Module: tf.keras

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras#aliases)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras#modules)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras#functions)

Implementation of the Keras API meant to be a high-level API for TensorFlow.

Aliases:

* Module tf.compat.v2.keras
* Module tf.keras

Defined in [python/keras/api/\_v2/keras/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/__init__.py).

Detailed documentation and user guides are available at [keras.io](https://keras.io/).

Modules

[activations](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations) module: Built-in activation functions.

[applications](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications) module: Keras Applications are canned architectures with pre-trained weights.

[backend](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend) module: Keras backend API.

[callbacks](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks) module: Callbacks: utilities called at certain points during model training.

[constraints](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints) module: Constraints: functions that impose constraints on weight values.

[datasets](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets) module: Keras built-in datasets.

[estimator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/estimator) module: Keras estimator API.

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental) module: Public API for tf.keras.experimental namespace.

[initializers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers) module: Keras initializer serialization / deserialization.

[layers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers) module: Keras layers API.

[losses](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses) module: Built-in loss functions.

[metrics](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics) module: Built-in metrics.

[mixed\_precision](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision) module: Public API for tf.keras.mixed\_precision namespace.

[models](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models) module: Code for model cloning, plus model-related API entries.

[optimizers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers) module: Built-in optimizer classes.

[preprocessing](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing) module: Keras data preprocessing utils.

[regularizers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers) module: Built-in regularizers.

[utils](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils) module: Keras utilities.

[wrappers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers) module: Wrappers for Keras models, providing compatibility with other frameworks.

Classes

[class Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model): Model groups layers into an object with training and inference features.

[class Sequential](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential): Linear stack of layers.

Functions

[Input(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Input): Input() is used to instantiate a Keras tensor.

# tf.keras.Input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Input#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Input#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Input#used_in_the_tutorials)

Input() is used to instantiate a Keras tensor.

### Aliases:

* tf.compat.v1.keras.Input
* tf.compat.v1.keras.layers.Input
* tf.compat.v2.keras.Input
* tf.compat.v2.keras.layers.Input
* tf.keras.Input
* tf.keras.layers.Input

tf.keras.Input(  
    shape=None,  
    batch\_size=None,  
    name=None,  
    dtype=None,  
    sparse=False,  
    tensor=None,  
    \*\*kwargs  
)

Defined in [python/keras/engine/input\_layer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/engine/input_layer.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Using GPUs](https://www.tensorflow.org/beta/guide/using_gpu)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Load CSV with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/csv)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

A Keras tensor is a tensor object from the underlying backend (Theano or TensorFlow), which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model.

For instance, if a, b and c are Keras tensors, it becomes possible to do: model = Model(input=[a, b], output=c)

The added Keras attribute is: \_keras\_history: Last layer applied to the tensor. the entire layer graph is retrievable from that layer, recursively.

#### Arguments:

* **shape**: A shape tuple (integers), not including the batch size. For instance, shape=(32,)indicates that the expected input will be batches of 32-dimensional vectors.
* **batch\_size**: optional static batch size (integer).
* **name**: An optional name string for the layer. Should be unique in a model (do not reuse the same name twice). It will be autogenerated if it isn't provided.
* **dtype**: The data type expected by the input, as a string (float32, float64, int32...)
* **sparse**: A boolean specifying whether the placeholder to be created is sparse.
* **tensor**: Optional existing tensor to wrap into the Input layer. If set, the layer will not create a placeholder tensor.
* **\*\*kwargs**: deprecated **Arguments** support.

#### Returns:

A tensor.

#### Example:

# this is a logistic regression in Keras  
x = Input(shape=(32,))  
y = Dense(16, activation='softmax')(x)  
model = Model(x, y)

Note that even if eager execution is enabled, Input produces a symbolic tensor (i.e. a placeholder). This symbolic tensor can be used with other TensorFlow ops, as such:

x = Input(shape=(32,))  
y = tf.square(x)

#### Raises:

* **ValueError**: in case of invalid **Arguments**.

# tf.keras.Model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#top_of_page)
* [Class Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#class_model)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#used_in_the_tutorials)

## Class Model

Model groups layers into an object with training and inference features.

### Aliases:

* Class tf.compat.v1.keras.Model
* Class tf.compat.v1.keras.models.Model
* Class tf.compat.v2.keras.Model
* Class tf.compat.v2.keras.models.Model
* Class tf.keras.Model
* Class tf.keras.models.Model

Defined in [python/keras/engine/training.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/engine/training.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Using GPUs](https://www.tensorflow.org/beta/guide/using_gpu)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load CSV with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/csv)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

There are two ways to instantiate a Model:

1 - With the "functional API", where you start from Input, you chain layer calls to specify the model's forward pass, and finally you create your model from inputs and outputs:

import tensorflow as tf  
  
inputs = tf.keras.Input(shape=(3,))  
x = tf.keras.layers.Dense(4, activation=tf.nn.relu)(inputs)  
outputs = tf.keras.layers.Dense(5, activation=tf.nn.softmax)(x)  
model = tf.keras.Model(inputs=inputs, outputs=outputs)

2 - By subclassing the Model class: in that case, you should define your layers in \_\_init\_\_ and you should implement the model's forward pass in call.

import tensorflow as tf  
  
class MyModel(tf.keras.Model):  
  
  def \_\_init\_\_(self):  
    super(MyModel, self).\_\_init\_\_()  
    self.dense1 = tf.keras.layers.Dense(4, activation=tf.nn.relu)  
    self.dense2 = tf.keras.layers.Dense(5, activation=tf.nn.softmax)  
  
  def call(self, inputs):  
    x = self.dense1(inputs)  
    return self.dense2(x)  
  
model = MyModel()

If you subclass Model, you can optionally have a training argument (boolean) in call, which you can use to specify a different behavior in training and inference:

import tensorflow as tf  
  
class MyModel(tf.keras.Model):  
  
  def \_\_init\_\_(self):  
    super(MyModel, self).\_\_init\_\_()  
    self.dense1 = tf.keras.layers.Dense(4, activation=tf.nn.relu)  
    self.dense2 = tf.keras.layers.Dense(5, activation=tf.nn.softmax)  
    self.dropout = tf.keras.layers.Dropout(0.5)  
  
  def call(self, inputs, training=False):  
    x = self.dense1(inputs)  
    if training:  
      x = self.dropout(x, training=training)  
    return self.dense2(x)  
  
model = MyModel()

## \_\_init\_\_

\_\_init\_\_(  
    \*args,  
    \*\*kwargs  
)

## Properties

### input\_spec

Gets the network's input specs.

#### Returns:

A list of InputSpec instances (one per input to the model) or a single instance if the model has only one input.

### layers

### metrics\_names

Returns the model's display labels for all outputs.

### run\_eagerly

Settable attribute indicating whether the model should run eagerly.

Running eagerly means that your model will be run step by step, like Python code. Your model might run slower, but it should become easier for you to debug it by stepping into individual layer calls.

By default, we will attempt to compile your model to a static graph to deliver the best execution performance.

#### Returns:

Boolean, whether the model should run eagerly.

### sample\_weights

### state\_updates

Returns the updates from all layers that are stateful.

This is useful for separating training updates and state updates, e.g. when we need to update a layer's internal state during prediction.

#### Returns:

A list of update ops.

### stateful

## Methods

### compile

compile(  
    optimizer,  
    loss=None,  
    metrics=None,  
    loss\_weights=None,  
    sample\_weight\_mode=None,  
    weighted\_metrics=None,  
    target\_tensors=None,  
    distribute=None,  
    \*\*kwargs  
)

Configures the model for training.

#### Arguments:

* **optimizer**: String (name of optimizer) or optimizer instance. See [tf.keras.optimizers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers).
* **loss**: String (name of objective function), objective function or [tf.losses.Loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss) instance. See [tf.losses](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses). If the model has multiple outputs, you can use a different loss on each output by passing a dictionary or a list of losses. The loss value that will be minimized by the model will then be the sum of all individual losses.
* **metrics**: List of metrics to be evaluated by the model during training and testing. Typically you will use metrics=['accuracy']. To specify different metrics for different outputs of a multi-output model, you could also pass a dictionary, such as metrics={'output\_a': 'accuracy', 'output\_b': ['accuracy', 'mse']}. You can also pass a list (len = len(outputs)) of lists of metrics such as metrics=[['accuracy'], ['accuracy', 'mse']] or metrics=['accuracy', ['accuracy', 'mse']].
* **loss\_weights**: Optional list or dictionary specifying scalar coefficients (Python floats) to weight the loss contributions of different model outputs. The loss value that will be minimized by the model will then be the weighted sum of all individual losses, weighted by the loss\_weightscoefficients. If a list, it is expected to have a 1:1 mapping to the model's outputs. If a tensor, it is expected to map output names (strings) to scalar coefficients.
* **sample\_weight\_mode**: If you need to do timestep-wise sample weighting (2D weights), set this to "temporal". None defaults to sample-wise weights (1D). If the model has multiple outputs, you can use a different sample\_weight\_mode on each output by passing a dictionary or a list of modes.
* **weighted\_metrics**: List of metrics to be evaluated and weighted by sample\_weight or class\_weight during training and testing.
* **target\_tensors**: By default, Keras will create placeholders for the model's target, which will be fed with the target data during training. If instead you would like to use your own target tensors (in turn, Keras will not expect external Numpy data for these targets at training time), you can specify them via the target\_tensors argument. It can be a single tensor (for a single-output model), a list of tensors, or a dict mapping output names to target tensors.
* **distribute**: NOT SUPPORTED IN TF 2.0, please create and compile the model under distribution strategy scope instead of passing it to compile.
* **\*\*kwargs**: Any additional **Arguments**.

#### Raises:

* **ValueError**: In case of invalid **Arguments** for optimizer, loss, metrics or sample\_weight\_mode.

### evaluate

evaluate(  
    x=None,  
    y=None,  
    batch\_size=None,  
    verbose=1,  
    sample\_weight=None,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False  
)

Returns the loss value & metrics values for the model in test mode.

Computation is done in batches.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
  + A generator or keras.utils.Sequence instance.
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset, dataset iterator, generator or keras.utils.Sequence instance, y should not be specified (since targets will be obtained from the iterator/dataset).
* **batch\_size**: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32. Do not specify the batch\_size is your data is in the form of symbolic tensors, dataset, dataset iterators, generators, or keras.utils.Sequence instances (since they generate batches).
* **verbose**: 0 or 1. Verbosity mode. 0 = silent, 1 = progress bar.
* **sample\_weight**: Optional Numpy array of weights for the test samples, used for weighting the loss function. You can either pass a flat (1D) Numpy array with the same length as the input samples (1:1 mapping between weights and samples), or in the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specifysample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset or a dataset iterator, instead pass sample weights as the third element of x.
* **steps**: Integer or None. Total number of steps (batches of samples) before declaring the evaluation round finished. Ignored with the default value of None. If x is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and steps is None, 'evaluate' will run until the dataset is exhausted.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during evaluation. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: Integer. Used for generator or keras.utils.Sequence input only. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use\_multiprocessing will default toFalse. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.

#### Returns:

Scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: in case of invalid **Arguments**.

### evaluate\_generator

evaluate\_generator(  
    generator,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    verbose=0  
)

Evaluates the model on a data generator.

The generator should return the same kind of data as accepted by test\_on\_batch.

#### Arguments:

* **generator**: Generator yielding tuples (inputs, targets) or (inputs, targets, sample\_weights) or an instance of keras.utils.Sequence object in order to avoid duplicate data when using multiprocessing.
* **steps**: Total number of steps (batches of samples) to yield from generator before stopping. Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during evaluation. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: maximum size for the generator queue
* **workers**: Integer. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. If True, use process-based threading. If unspecified, use\_multiprocessing will default to False. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **verbose**: Verbosity mode, 0 or 1.

#### Returns:

Scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: in case of invalid **Arguments**.

#### Raises:

* **ValueError**: In case the generator yields data in an invalid format.

### fit

fit(  
    x=None,  
    y=None,  
    batch\_size=None,  
    epochs=1,  
    verbose=1,  
    callbacks=None,  
    validation\_split=0.0,  
    validation\_data=None,  
    shuffle=True,  
    class\_weight=None,  
    sample\_weight=None,  
    initial\_epoch=0,  
    steps\_per\_epoch=None,  
    validation\_steps=None,  
    validation\_freq=1,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    \*\*kwargs  
)

Trains the model for a fixed number of epochs (iterations on a dataset).

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator. Should return a tuple of either (inputs, targets) or (inputs, targets, sample\_weights).
  + A generator or keras.utils.Sequence returning (inputs, targets) or (inputs, targets, sample weights).
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset, dataset iterator, generator, or keras.utils.Sequence instance, y should not be specified (since targets will be obtained from x).
* **batch\_size**: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32. Do not specify the batch\_size if your data is in the form of symbolic tensors, dataset, dataset iterators, generators, or keras.utils.Sequence instances (since they generate batches).
* **epochs**: Integer. Number of epochs to train the model. An epoch is an iteration over the entire xand y data provided. Note that in conjunction with initial\_epoch, epochs is to be understood as "final epoch". The model is not trained for a number of iterations given by epochs, but merely until the epoch of index epochs is reached.
* **verbose**: 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch. Note that the progress bar is not particularly useful when logged to a file, so verbose=2 is recommended when not running interactively (eg, in a production environment).
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during training. See [tf.keras.callbacks](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks).
* **validation\_split**: Float between 0 and 1. Fraction of the training data to be used as validation data. The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch. The validation data is selected from the last samples in the x and y data provided, before shuffling. This argument is not supported when x is a dataset, dataset iterator, generator orkeras.utils.Sequence instance.
* **validation\_data**: Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data. validation\_data will override validation\_split. validation\_data could be:
  + tuple (x\_val, y\_val) of Numpy arrays or tensors
  + tuple (x\_val, y\_val, val\_sample\_weights) of Numpy arrays
  + dataset or a dataset iterator For the first two cases, batch\_size must be provided. For the last case, validation\_steps must be provided.
* **shuffle**: Boolean (whether to shuffle the training data before each epoch) or str (for 'batch'). 'batch' is a special option for dealing with the limitations of HDF5 data; it shuffles in batch-sized chunks. Has no effect when steps\_per\_epoch is not None.
* **class\_weight**: Optional dictionary mapping class indices (integers) to a weight (float) value, used for weighting the loss function (during training only). This can be useful to tell the model to "pay more attention" to samples from an under-represented class.
* **sample\_weight**: Optional Numpy array of weights for the training samples, used for weighting the loss function (during training only). You can either pass a flat (1D) Numpy array with the same length as the input samples (1:1 mapping between weights and samples), or in the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specifysample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset, dataset iterator, generator, or keras.utils.Sequence instance, instead provide the sample\_weights as the third element of x.
* **initial\_epoch**: Integer. Epoch at which to start training (useful for resuming a previous training run).
* **steps\_per\_epoch**: Integer or None. Total number of steps (batches of samples) before declaring one epoch finished and starting the next epoch. When training with input tensors such as TensorFlow data tensors, the default None is equal to the number of samples in your dataset divided by the batch size, or 1 if that cannot be determined. If x is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and 'steps\_per\_epoch' is None, the epoch will run until the input dataset is exhausted.
* **validation\_steps**: Only relevant if validation\_data is provided and is a dataset or dataset iterator. Total number of steps (batches of samples) to draw before stopping when performing validation at the end of every epoch. If validation\_data is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and 'validation\_steps' is None, validation will run until the validation\_data dataset is exhausted.
* **validation\_freq**: Only relevant if validation data is provided. Integer or collections.Container instance (e.g. list, tuple, etc.). If an integer, specifies how many training epochs to run before a new validation run is performed, e.g. validation\_freq=2 runs validation every 2 epochs. If a Container, specifies the epochs on which to run validation, e.g. validation\_freq=[1, 2, 10] runs validation at the end of the 1st, 2nd, and 10th epochs.
* **max\_queue\_size**: Integer. Used for generator or keras.utils.Sequence input only. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use\_multiprocessing will default toFalse. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **\*\*kwargs**: Used for backwards compatibility.

#### Returns:

A History object. Its History.history attribute is a record of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values (if applicable).

#### Raises:

* **RuntimeError**: If the model was never compiled.
* **ValueError**: In case of mismatch between the provided input data and what the model expects.

### fit\_generator

fit\_generator(  
    generator,  
    steps\_per\_epoch=None,  
    epochs=1,  
    verbose=1,  
    callbacks=None,  
    validation\_data=None,  
    validation\_steps=None,  
    validation\_freq=1,  
    class\_weight=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    shuffle=True,  
    initial\_epoch=0  
)

Fits the model on data yielded batch-by-batch by a Python generator.

The generator is run in parallel to the model, for efficiency. For instance, this allows you to do real-time data augmentation on images on CPU in parallel to training your model on GPU.

The use of keras.utils.Sequence guarantees the ordering and guarantees the single use of every input per epoch when using use\_multiprocessing=True.

#### Arguments:

* **generator**: A generator or an instance of Sequence (keras.utils.Sequence) object in order to avoid duplicate data when using multiprocessing. The output of the generator must be either
  + a tuple (inputs, targets)
  + a tuple (inputs, targets, sample\_weights). This tuple (a single output of the generator) makes a single batch. Therefore, all arrays in this tuple must have the same length (equal to the size of this batch). Different batches may have different sizes. For example, the last batch of the epoch is commonly smaller than the others, if the size of the dataset is not divisible by the batch size. The generator is expected to loop over its data indefinitely. An epoch finishes when steps\_per\_epoch batches have been seen by the model.
* **steps\_per\_epoch**: Total number of steps (batches of samples) to yield from generatorbefore declaring one epoch finished and starting the next epoch. It should typically be equal to the number of samples of your dataset divided by the batch size. Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
* **epochs**: Integer, total number of iterations on the data.
* **verbose**: Verbosity mode, 0, 1, or 2.
* **callbacks**: List of callbacks to be called during training.
* **validation\_data**: This can be either
  + a generator for the validation data
  + a tuple (inputs, targets)
  + a tuple (inputs, targets, sample\_weights).
* **validation\_steps**: Only relevant if validation\_data is a generator. Total number of steps (batches of samples) to yield from generator before stopping. Optional for Sequence: if unspecified, will use the len(validation\_data) as a number of steps.
* **validation\_freq**: Only relevant if validation data is provided. Integer or collections.Container instance (e.g. list, tuple, etc.). If an integer, specifies how many training epochs to run before a new validation run is performed, e.g. validation\_freq=2 runs validation every 2 epochs. If a Container, specifies the epochs on which to run validation, e.g. validation\_freq=[1, 2, 10] runs validation at the end of the 1st, 2nd, and 10th epochs.
* **class\_weight**: Dictionary mapping class indices to a weight for the class.
* **max\_queue\_size**: Integer. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. If True, use process-based threading. If unspecified, use\_multiprocessing will default to False. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **shuffle**: Boolean. Whether to shuffle the order of the batches at the beginning of each epoch. Only used with instances of Sequence (keras.utils.Sequence). Has no effect when steps\_per\_epoch is not None.
* **initial\_epoch**: Epoch at which to start training (useful for resuming a previous training run)

#### Returns:

A History object.

#### Example:

    def generate\_arrays\_from\_file(path):  
        while 1:  
            f = open(path)  
            for line in f:  
                # create numpy arrays of input data  
                # and labels, from each line in the file  
                x1, x2, y = process\_line(line)  
                yield ({'input\_1': x1, 'input\_2': x2}, {'output': y})  
            f.close()  
  
    model.fit\_generator(generate\_arrays\_from\_file('/my\_file.txt'),  
                        steps\_per\_epoch=10000, epochs=10)

Raises: ValueError: In case the generator yields data in an invalid format.

### get\_layer

get\_layer(  
    name=None,  
    index=None  
)

Retrieves a layer based on either its name (unique) or index.

If name and index are both provided, index will take precedence. Indices are based on order of horizontal graph traversal (bottom-up).

#### Arguments:

* **name**: String, name of layer.
* **index**: Integer, index of layer.

#### Returns:

A layer instance.

#### Raises:

* **ValueError**: In case of invalid layer name or index.

### load\_weights

load\_weights(  
    filepath,  
    by\_name=False  
)

Loads all layer weights, either from a TensorFlow or an HDF5 file.

### predict

predict(  
    x,  
    batch\_size=None,  
    verbose=0,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False  
)

Generates output predictions for the input samples.

Computation is done in batches.

#### Arguments:

* **x**: Input samples. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
  + A generator or keras.utils.Sequence instance.
* **batch\_size**: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32. Do not specify the batch\_size is your data is in the form of symbolic tensors, dataset, dataset iterators, generators, or keras.utils.Sequence instances (since they generate batches).
* **verbose**: Verbosity mode, 0 or 1.
* **steps**: Total number of steps (batches of samples) before declaring the prediction round finished. Ignored with the default value of None. If x is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and steps is None, predict will run until the input dataset is exhausted.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during prediction. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: Integer. Used for generator or keras.utils.Sequence input only. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use\_multiprocessing will default toFalse. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.

#### Returns:

Numpy array(s) of predictions.

#### Raises:

* **ValueError**: In case of mismatch between the provided input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

### predict\_generator

predict\_generator(  
    generator,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    verbose=0  
)

Generates predictions for the input samples from a data generator.

The generator should return the same kind of data as accepted by predict\_on\_batch.

#### Arguments:

* **generator**: Generator yielding batches of input samples or an instance of keras.utils.Sequence object in order to avoid duplicate data when using multiprocessing.
* **steps**: Total number of steps (batches of samples) to yield from generator before stopping. Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during prediction. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: Maximum size for the generator queue.
* **workers**: Integer. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. If True, use process-based threading. If unspecified, use\_multiprocessing will default to False. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **verbose**: verbosity mode, 0 or 1.

#### Returns:

Numpy array(s) of predictions.

#### Raises:

* **ValueError**: In case the generator yields data in an invalid format.

### predict\_on\_batch

predict\_on\_batch(x)

Returns predictions for a single batch of samples.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.

#### Returns:

Numpy array(s) of predictions.

#### Raises:

* **ValueError**: In case of mismatch between given number of inputs and expectations of the model.

### reset\_metrics

reset\_metrics()

Resets the state of metrics.

### reset\_states

reset\_states()

### save

save(  
    filepath,  
    overwrite=True,  
    include\_optimizer=True,  
    save\_format=None  
)

Saves the model to Tensorflow SavedModel or a single HDF5 file.

#### The savefile includes:

* The model architecture, allowing to re-instantiate the model.
* The model weights.
* The state of the optimizer, allowing to resume training exactly where you left off.

This allows you to save the entirety of the state of a model in a single file.

Saved models can be reinstantiated via keras.models.load\_model. The model returned by load\_model is a compiled model ready to be used (unless the saved model was never compiled in the first place).

#### Arguments:

* **filepath**: String, path to SavedModel or H5 file to save the model.
* **overwrite**: Whether to silently overwrite any existing file at the target location, or provide the user with a manual prompt.
* **include\_optimizer**: If True, save optimizer's state together.
* **save\_format**: Either 'tf' or 'h5', indicating whether to save the model to Tensorflow SavedModel or HDF5. The default is currently 'h5', but will switch to 'tf' in TensorFlow 2.0. The 'tf' option is currently disabled (use [tf.keras.experimental.export\_saved\_model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model) instead).

#### Example:

from keras.models import load\_model  
  
model.save('my\_model.h5')  # creates a HDF5 file 'my\_model.h5'  
del model  # deletes the existing model  
  
# returns a compiled model  
# identical to the previous one  
model = load\_model('my\_model.h5')

### save\_weights

save\_weights(  
    filepath,  
    overwrite=True,  
    save\_format=None  
)

Saves all layer weights.

Either saves in HDF5 or in TensorFlow format based on the save\_format argument.

When saving in HDF5 format, the weight file has: - layer\_names (attribute), a list of strings (ordered names of model layers). - For every layer, a group named layer.name - For every such layer group, a group attribute weight\_names, a list of strings (ordered names of weights tensor of the layer). - For every weight in the layer, a dataset storing the weight value, named after the weight tensor.

When saving in TensorFlow format, all objects referenced by the network are saved in the same format as [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint), including any Layer instances or Optimizer instances assigned to object attributes. For networks constructed from inputs and outputs using tf.keras.Model(inputs, outputs), Layer instances used by the network are tracked/saved automatically. For user-defined classes which inherit from [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model), Layer instances must be assigned to object attributes, typically in the constructor. See the documentation of [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) and [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model)for details.

While the formats are the same, do not mix save\_weights and [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint). Checkpoints saved by Model.save\_weights should be loaded using Model.load\_weights. Checkpoints saved using [tf.train.Checkpoint.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#save) should be restored using the corresponding[tf.train.Checkpoint.restore](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#restore). Prefer [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) over save\_weights for training checkpoints.

The TensorFlow format matches objects and variables by starting at a root object, self for save\_weights, and greedily matching attribute names. For Model.save this is the Model, and for Checkpoint.save this is the Checkpoint even if the Checkpoint has a model attached. This means saving a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) using save\_weights and loading into a [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint)with a Model attached (or vice versa) will not match the Model's variables. See the [guide to training checkpoints](https://www.tensorflow.org/alpha/guide/checkpoints) for details on the TensorFlow format.

#### Arguments:

* **filepath**: String, path to the file to save the weights to. When saving in TensorFlow format, this is the prefix used for checkpoint files (multiple files are generated). Note that the '.h5' suffix causes weights to be saved in HDF5 format.
* **overwrite**: Whether to silently overwrite any existing file at the target location, or provide the user with a manual prompt.
* **save\_format**: Either 'tf' or 'h5'. A filepath ending in '.h5' or '.keras' will default to HDF5 if save\_format is None. Otherwise None defaults to 'tf'.

#### Raises:

* **ImportError**: If h5py is not available when attempting to save in HDF5 format.
* **ValueError**: For invalid/unknown format **Arguments**.

### summary

summary(  
    line\_length=None,  
    positions=None,  
    print\_fn=None  
)

Prints a string summary of the network.

#### Arguments:

* **line\_length**: Total length of printed lines (e.g. set this to adapt the display to different terminal window sizes).
* **positions**: Relative or absolute positions of log elements in each line. If not provided, defaults to [.33, .55, .67, 1.].
* **print\_fn**: Print function to use. Defaults to print. It will be called on each line of the summary. You can set it to a custom function in order to capture the string summary.

#### Raises:

* **ValueError**: if summary() is called before the model is built.

### test\_on\_batch

test\_on\_batch(  
    x,  
    y=None,  
    sample\_weight=None,  
    reset\_metrics=True  
)

Test the model on a single batch of samples.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset or a dataset iterator, y should not be specified (since targets will be obtained from the iterator).
* **sample\_weight**: Optional array of the same length as x, containing weights to apply to the model's loss for each sample. In the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specify sample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset or a dataset iterator.
* **reset\_metrics**: If True, the metrics returned will be only for this batch. If False, the metrics will be statefully accumulated across batches.

#### Returns:

Scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: In case of invalid user-provided **Arguments**.

### to\_json

to\_json(\*\*kwargs)

Returns a JSON string containing the network configuration.

To load a network from a JSON save file, use keras.models.model\_from\_json(json\_string, custom\_objects={}).

#### Arguments:

* **\*\*kwargs**: Additional keyword **Arguments** to be passed to json.dumps().

#### Returns:

A JSON string.

### to\_yaml

to\_yaml(\*\*kwargs)

Returns a yaml string containing the network configuration.

To load a network from a yaml save file, use keras.models.model\_from\_yaml(yaml\_string, custom\_objects={}).

custom\_objects should be a dictionary mapping the names of custom losses / layers / etc to the corresponding functions / classes.

#### Arguments:

* **\*\*kwargs**: Additional keyword **Arguments** to be passed to yaml.dump().

#### Returns:

A YAML string.

#### Raises:

* **ImportError**: if yaml module is not found.

### train\_on\_batch

train\_on\_batch(  
    x,  
    y=None,  
    sample\_weight=None,  
    class\_weight=None,  
    reset\_metrics=True  
)

Runs a single gradient update on a single batch of data.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset or a dataset iterator, y should not be specified (since targets will be obtained from the iterator).
* **sample\_weight**: Optional array of the same length as x, containing weights to apply to the model's loss for each sample. In the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specify sample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset or a dataset iterator.
* **class\_weight**: Optional dictionary mapping class indices (integers) to a weight (float) to apply to the model's loss for the samples from this class during training. This can be useful to tell the model to "pay more attention" to samples from an under-represented class.
* **reset\_metrics**: If True, the metrics returned will be only for this batch. If False, the metrics will be statefully accumulated across batches.

#### Returns:

Scalar training loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: In case of invalid user-provided **Arguments**.

# tf.keras.Sequential

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential#top_of_page)
* [Class Sequential](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential#class_sequential)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential#used_in_the_tutorials)

## Class Sequential

Linear stack of layers.

Inherits From: [Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model)

### Aliases:

* Class tf.compat.v1.keras.Sequential
* Class tf.compat.v1.keras.models.Sequential
* Class tf.compat.v2.keras.Sequential
* Class tf.compat.v2.keras.models.Sequential
* Class tf.keras.Sequential
* Class tf.keras.models.Sequential

Defined in [python/keras/engine/sequential.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/engine/sequential.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Keras custom callbacks](https://www.tensorflow.org/beta/guide/keras/custom_callback)
* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)
* [Convolutional Neural Networks](https://www.tensorflow.org/beta/tutorials/images/intro_to_cnns)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom layers](https://www.tensorflow.org/beta/tutorials/eager/custom_layers)
* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Explore overfitting and underfitting](https://www.tensorflow.org/beta/tutorials/keras/overfit_and_underfit)
* [Get started with TensorFlow 2.0 for beginners](https://www.tensorflow.org/beta/tutorials/quickstart/beginner)
* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Load text with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/text)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Multi-worker Training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_keras)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Regression: Predict fuel efficiency](https://www.tensorflow.org/beta/tutorials/keras/basic_regression)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)
* [Text classification of movie reviews with Keras and TensorFlow Hub](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification_with_tfhub)
* [Text classification with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_classification_rnn)
* [Text classification with movie reviews](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Train your first neural network: basic classification](https://www.tensorflow.org/beta/tutorials/keras/basic_classification)
* [Transfer Learning Using Pretrained ConvNets](https://www.tensorflow.org/beta/tutorials/images/transfer_learning)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [Word embeddings](https://www.tensorflow.org/beta/tutorials/text/word_embeddings)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

#### Arguments:

* **layers**: list of layers to add to the model.

#### Example:

# Optionally, the first layer can receive an `input\_shape` argument:  
model = Sequential()  
model.add(Dense(32, input\_shape=(500,)))  
# Afterwards, we do automatic shape inference:  
model.add(Dense(32))  
  
# This is identical to the following:  
model = Sequential()  
model.add(Dense(32, input\_dim=500))  
  
# And to the following:  
model = Sequential()  
model.add(Dense(32, batch\_input\_shape=(None, 500)))  
  
# Note that you can also omit the `input\_shape` argument:  
# In that case the model gets built the first time you call `fit` (or other  
# training and evaluation methods).  
model = Sequential()  
model.add(Dense(32))  
model.add(Dense(32))  
model.compile(optimizer=optimizer, loss=loss)  
# This builds the model for the first time:  
model.fit(x, y, batch\_size=32, epochs=10)  
  
# Note that when using this delayed-build pattern (no input shape specified),  
# the model doesn't have any weights until the first call  
# to a training/evaluation method (since it isn't yet built):  
model = Sequential()  
model.add(Dense(32))  
model.add(Dense(32))  
model.weights  # returns []  
  
# Whereas if you specify the input shape, the model gets built continuously  
# as you are adding layers:  
model = Sequential()  
model.add(Dense(32, input\_shape=(500,)))  
model.add(Dense(32))  
model.weights  # returns list of length 4  
  
# When using the delayed-build pattern (no input shape specified), you can  
# choose to manually build your model by calling `build(batch\_input\_shape)`:  
model = Sequential()  
model.add(Dense(32))  
model.add(Dense(32))  
model.build((None, 500))  
model.weights  # returns list of length 4

## \_\_init\_\_

\_\_init\_\_(  
    layers=None,  
    name=None  
)

## Properties

### input\_spec

### layers

### metrics\_names

Returns the model's display labels for all outputs.

### run\_eagerly

Settable attribute indicating whether the model should run eagerly.

Running eagerly means that your model will be run step by step, like Python code. Your model might run slower, but it should become easier for you to debug it by stepping into individual layer calls.

By default, we will attempt to compile your model to a static graph to deliver the best execution performance.

#### Returns:

Boolean, whether the model should run eagerly.

### sample\_weights

### state\_updates

Returns the updates from all layers that are stateful.

This is useful for separating training updates and state updates, e.g. when we need to update a layer's internal state during prediction.

#### Returns:

A list of update ops.

### stateful

## Methods

### add

add(layer)

Adds a layer instance on top of the layer stack.

#### Arguments:

* **layer**: layer instance.

#### Raises:

* **TypeError**: If layer is not a layer instance.
* **ValueError**: In case the layer argument does not know its input shape.
* **ValueError**: In case the layer argument has multiple output tensors, or is already connected somewhere else (forbidden in Sequential models).

### compile

compile(  
    optimizer,  
    loss=None,  
    metrics=None,  
    loss\_weights=None,  
    sample\_weight\_mode=None,  
    weighted\_metrics=None,  
    target\_tensors=None,  
    distribute=None,  
    \*\*kwargs  
)

Configures the model for training.

#### Arguments:

* **optimizer**: String (name of optimizer) or optimizer instance. See [tf.keras.optimizers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers).
* **loss**: String (name of objective function), objective function or [tf.losses.Loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss) instance. See [tf.losses](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses). If the model has multiple outputs, you can use a different loss on each output by passing a dictionary or a list of losses. The loss value that will be minimized by the model will then be the sum of all individual losses.
* **metrics**: List of metrics to be evaluated by the model during training and testing. Typically you will use metrics=['accuracy']. To specify different metrics for different outputs of a multi-output model, you could also pass a dictionary, such as metrics={'output\_a': 'accuracy', 'output\_b': ['accuracy', 'mse']}. You can also pass a list (len = len(outputs)) of lists of metrics such as metrics=[['accuracy'], ['accuracy', 'mse']] or metrics=['accuracy', ['accuracy', 'mse']].
* **loss\_weights**: Optional list or dictionary specifying scalar coefficients (Python floats) to weight the loss contributions of different model outputs. The loss value that will be minimized by the model will then be the weighted sum of all individual losses, weighted by the loss\_weightscoefficients. If a list, it is expected to have a 1:1 mapping to the model's outputs. If a tensor, it is expected to map output names (strings) to scalar coefficients.
* **sample\_weight\_mode**: If you need to do timestep-wise sample weighting (2D weights), set this to "temporal". None defaults to sample-wise weights (1D). If the model has multiple outputs, you can use a different sample\_weight\_mode on each output by passing a dictionary or a list of modes.
* **weighted\_metrics**: List of metrics to be evaluated and weighted by sample\_weight or class\_weight during training and testing.
* **target\_tensors**: By default, Keras will create placeholders for the model's target, which will be fed with the target data during training. If instead you would like to use your own target tensors (in turn, Keras will not expect external Numpy data for these targets at training time), you can specify them via the target\_tensors argument. It can be a single tensor (for a single-output model), a list of tensors, or a dict mapping output names to target tensors.
* **distribute**: NOT SUPPORTED IN TF 2.0, please create and compile the model under distribution strategy scope instead of passing it to compile.
* **\*\*kwargs**: Any additional **Arguments**.

#### Raises:

* **ValueError**: In case of invalid **Arguments** for optimizer, loss, metrics or sample\_weight\_mode.

### evaluate

evaluate(  
    x=None,  
    y=None,  
    batch\_size=None,  
    verbose=1,  
    sample\_weight=None,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False  
)

Returns the loss value & metrics values for the model in test mode.

Computation is done in batches.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
  + A generator or keras.utils.Sequence instance.
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset, dataset iterator, generator or keras.utils.Sequence instance, y should not be specified (since targets will be obtained from the iterator/dataset).
* **batch\_size**: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32. Do not specify the batch\_size is your data is in the form of symbolic tensors, dataset, dataset iterators, generators, or keras.utils.Sequence instances (since they generate batches).
* **verbose**: 0 or 1. Verbosity mode. 0 = silent, 1 = progress bar.
* **sample\_weight**: Optional Numpy array of weights for the test samples, used for weighting the loss function. You can either pass a flat (1D) Numpy array with the same length as the input samples (1:1 mapping between weights and samples), or in the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specifysample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset or a dataset iterator, instead pass sample weights as the third element of x.
* **steps**: Integer or None. Total number of steps (batches of samples) before declaring the evaluation round finished. Ignored with the default value of None. If x is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and steps is None, 'evaluate' will run until the dataset is exhausted.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during evaluation. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: Integer. Used for generator or keras.utils.Sequence input only. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use\_multiprocessing will default toFalse. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.

#### Returns:

Scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: in case of invalid **Arguments**.

### evaluate\_generator

evaluate\_generator(  
    generator,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    verbose=0  
)

Evaluates the model on a data generator.

The generator should return the same kind of data as accepted by test\_on\_batch.

#### Arguments:

* **generator**: Generator yielding tuples (inputs, targets) or (inputs, targets, sample\_weights) or an instance of keras.utils.Sequence object in order to avoid duplicate data when using multiprocessing.
* **steps**: Total number of steps (batches of samples) to yield from generator before stopping. Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during evaluation. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: maximum size for the generator queue
* **workers**: Integer. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. If True, use process-based threading. If unspecified, use\_multiprocessing will default to False. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **verbose**: Verbosity mode, 0 or 1.

#### Returns:

Scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: in case of invalid **Arguments**.

#### Raises:

* **ValueError**: In case the generator yields data in an invalid format.

### fit

fit(  
    x=None,  
    y=None,  
    batch\_size=None,  
    epochs=1,  
    verbose=1,  
    callbacks=None,  
    validation\_split=0.0,  
    validation\_data=None,  
    shuffle=True,  
    class\_weight=None,  
    sample\_weight=None,  
    initial\_epoch=0,  
    steps\_per\_epoch=None,  
    validation\_steps=None,  
    validation\_freq=1,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    \*\*kwargs  
)

Trains the model for a fixed number of epochs (iterations on a dataset).

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator. Should return a tuple of either (inputs, targets) or (inputs, targets, sample\_weights).
  + A generator or keras.utils.Sequence returning (inputs, targets) or (inputs, targets, sample weights).
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset, dataset iterator, generator, or keras.utils.Sequence instance, y should not be specified (since targets will be obtained from x).
* **batch\_size**: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32. Do not specify the batch\_size if your data is in the form of symbolic tensors, dataset, dataset iterators, generators, or keras.utils.Sequence instances (since they generate batches).
* **epochs**: Integer. Number of epochs to train the model. An epoch is an iteration over the entire xand y data provided. Note that in conjunction with initial\_epoch, epochs is to be understood as "final epoch". The model is not trained for a number of iterations given by epochs, but merely until the epoch of index epochs is reached.
* **verbose**: 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch. Note that the progress bar is not particularly useful when logged to a file, so verbose=2 is recommended when not running interactively (eg, in a production environment).
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during training. See [tf.keras.callbacks](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks).
* **validation\_split**: Float between 0 and 1. Fraction of the training data to be used as validation data. The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch. The validation data is selected from the last samples in the x and y data provided, before shuffling. This argument is not supported when x is a dataset, dataset iterator, generator orkeras.utils.Sequence instance.
* **validation\_data**: Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data. validation\_data will override validation\_split. validation\_data could be:
  + tuple (x\_val, y\_val) of Numpy arrays or tensors
  + tuple (x\_val, y\_val, val\_sample\_weights) of Numpy arrays
  + dataset or a dataset iterator For the first two cases, batch\_size must be provided. For the last case, validation\_steps must be provided.
* **shuffle**: Boolean (whether to shuffle the training data before each epoch) or str (for 'batch'). 'batch' is a special option for dealing with the limitations of HDF5 data; it shuffles in batch-sized chunks. Has no effect when steps\_per\_epoch is not None.
* **class\_weight**: Optional dictionary mapping class indices (integers) to a weight (float) value, used for weighting the loss function (during training only). This can be useful to tell the model to "pay more attention" to samples from an under-represented class.
* **sample\_weight**: Optional Numpy array of weights for the training samples, used for weighting the loss function (during training only). You can either pass a flat (1D) Numpy array with the same length as the input samples (1:1 mapping between weights and samples), or in the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specifysample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset, dataset iterator, generator, or keras.utils.Sequence instance, instead provide the sample\_weights as the third element of x.
* **initial\_epoch**: Integer. Epoch at which to start training (useful for resuming a previous training run).
* **steps\_per\_epoch**: Integer or None. Total number of steps (batches of samples) before declaring one epoch finished and starting the next epoch. When training with input tensors such as TensorFlow data tensors, the default None is equal to the number of samples in your dataset divided by the batch size, or 1 if that cannot be determined. If x is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and 'steps\_per\_epoch' is None, the epoch will run until the input dataset is exhausted.
* **validation\_steps**: Only relevant if validation\_data is provided and is a dataset or dataset iterator. Total number of steps (batches of samples) to draw before stopping when performing validation at the end of every epoch. If validation\_data is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and 'validation\_steps' is None, validation will run until the validation\_data dataset is exhausted.
* **validation\_freq**: Only relevant if validation data is provided. Integer or collections.Container instance (e.g. list, tuple, etc.). If an integer, specifies how many training epochs to run before a new validation run is performed, e.g. validation\_freq=2 runs validation every 2 epochs. If a Container, specifies the epochs on which to run validation, e.g. validation\_freq=[1, 2, 10] runs validation at the end of the 1st, 2nd, and 10th epochs.
* **max\_queue\_size**: Integer. Used for generator or keras.utils.Sequence input only. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use\_multiprocessing will default toFalse. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **\*\*kwargs**: Used for backwards compatibility.

#### Returns:

A History object. Its History.history attribute is a record of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values (if applicable).

#### Raises:

* **RuntimeError**: If the model was never compiled.
* **ValueError**: In case of mismatch between the provided input data and what the model expects.

### fit\_generator

fit\_generator(  
    generator,  
    steps\_per\_epoch=None,  
    epochs=1,  
    verbose=1,  
    callbacks=None,  
    validation\_data=None,  
    validation\_steps=None,  
    validation\_freq=1,  
    class\_weight=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    shuffle=True,  
    initial\_epoch=0  
)

Fits the model on data yielded batch-by-batch by a Python generator.

The generator is run in parallel to the model, for efficiency. For instance, this allows you to do real-time data augmentation on images on CPU in parallel to training your model on GPU.

The use of keras.utils.Sequence guarantees the ordering and guarantees the single use of every input per epoch when using use\_multiprocessing=True.

#### Arguments:

* **generator**: A generator or an instance of Sequence (keras.utils.Sequence) object in order to avoid duplicate data when using multiprocessing. The output of the generator must be either
  + a tuple (inputs, targets)
  + a tuple (inputs, targets, sample\_weights). This tuple (a single output of the generator) makes a single batch. Therefore, all arrays in this tuple must have the same length (equal to the size of this batch). Different batches may have different sizes. For example, the last batch of the epoch is commonly smaller than the others, if the size of the dataset is not divisible by the batch size. The generator is expected to loop over its data indefinitely. An epoch finishes when steps\_per\_epoch batches have been seen by the model.
* **steps\_per\_epoch**: Total number of steps (batches of samples) to yield from generatorbefore declaring one epoch finished and starting the next epoch. It should typically be equal to the number of samples of your dataset divided by the batch size. Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
* **epochs**: Integer, total number of iterations on the data.
* **verbose**: Verbosity mode, 0, 1, or 2.
* **callbacks**: List of callbacks to be called during training.
* **validation\_data**: This can be either
  + a generator for the validation data
  + a tuple (inputs, targets)
  + a tuple (inputs, targets, sample\_weights).
* **validation\_steps**: Only relevant if validation\_data is a generator. Total number of steps (batches of samples) to yield from generator before stopping. Optional for Sequence: if unspecified, will use the len(validation\_data) as a number of steps.
* **validation\_freq**: Only relevant if validation data is provided. Integer or collections.Container instance (e.g. list, tuple, etc.). If an integer, specifies how many training epochs to run before a new validation run is performed, e.g. validation\_freq=2 runs validation every 2 epochs. If a Container, specifies the epochs on which to run validation, e.g. validation\_freq=[1, 2, 10] runs validation at the end of the 1st, 2nd, and 10th epochs.
* **class\_weight**: Dictionary mapping class indices to a weight for the class.
* **max\_queue\_size**: Integer. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. If True, use process-based threading. If unspecified, use\_multiprocessing will default to False. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **shuffle**: Boolean. Whether to shuffle the order of the batches at the beginning of each epoch. Only used with instances of Sequence (keras.utils.Sequence). Has no effect when steps\_per\_epoch is not None.
* **initial\_epoch**: Epoch at which to start training (useful for resuming a previous training run)

#### Returns:

A History object.

#### Example:

    def generate\_arrays\_from\_file(path):  
        while 1:  
            f = open(path)  
            for line in f:  
                # create numpy arrays of input data  
                # and labels, from each line in the file  
                x1, x2, y = process\_line(line)  
                yield ({'input\_1': x1, 'input\_2': x2}, {'output': y})  
            f.close()  
  
    model.fit\_generator(generate\_arrays\_from\_file('/my\_file.txt'),  
                        steps\_per\_epoch=10000, epochs=10)

Raises: ValueError: In case the generator yields data in an invalid format.

### get\_layer

get\_layer(  
    name=None,  
    index=None  
)

Retrieves a layer based on either its name (unique) or index.

If name and index are both provided, index will take precedence. Indices are based on order of horizontal graph traversal (bottom-up).

#### Arguments:

* **name**: String, name of layer.
* **index**: Integer, index of layer.

#### Returns:

A layer instance.

#### Raises:

* **ValueError**: In case of invalid layer name or index.

### load\_weights

load\_weights(  
    filepath,  
    by\_name=False  
)

Loads all layer weights, either from a TensorFlow or an HDF5 file.

### pop

pop()

Removes the last layer in the model.

#### Raises:

* **TypeError**: if there are no layers in the model.

### predict

predict(  
    x,  
    batch\_size=None,  
    verbose=0,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False  
)

Generates output predictions for the input samples.

Computation is done in batches.

#### Arguments:

* **x**: Input samples. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
  + A generator or keras.utils.Sequence instance.
* **batch\_size**: Integer or None. Number of samples per gradient update. If unspecified, batch\_size will default to 32. Do not specify the batch\_size is your data is in the form of symbolic tensors, dataset, dataset iterators, generators, or keras.utils.Sequence instances (since they generate batches).
* **verbose**: Verbosity mode, 0 or 1.
* **steps**: Total number of steps (batches of samples) before declaring the prediction round finished. Ignored with the default value of None. If x is a [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator, and steps is None, predict will run until the input dataset is exhausted.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during prediction. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: Integer. Used for generator or keras.utils.Sequence input only. Maximum size for the generator queue. If unspecified, max\_queue\_size will default to 10.
* **workers**: Integer. Used for generator or keras.utils.Sequence input only. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. Used for generator or keras.utils.Sequence input only. If True, use process-based threading. If unspecified, use\_multiprocessing will default toFalse. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.

#### Returns:

Numpy array(s) of predictions.

#### Raises:

* **ValueError**: In case of mismatch between the provided input data and the model's expectations, or in case a stateful model receives a number of samples that is not a multiple of the batch size.

### predict\_classes

predict\_classes(  
    x,  
    batch\_size=32,  
    verbose=0  
)

Generate class predictions for the input samples.

The input samples are processed batch by batch.

#### Arguments:

* **x**: input data, as a Numpy array or list of Numpy arrays (if the model has multiple inputs).
* **batch\_size**: integer.
* **verbose**: verbosity mode, 0 or 1.

#### Returns:

A numpy array of class predictions.

### predict\_generator

predict\_generator(  
    generator,  
    steps=None,  
    callbacks=None,  
    max\_queue\_size=10,  
    workers=1,  
    use\_multiprocessing=False,  
    verbose=0  
)

Generates predictions for the input samples from a data generator.

The generator should return the same kind of data as accepted by predict\_on\_batch.

#### Arguments:

* **generator**: Generator yielding batches of input samples or an instance of keras.utils.Sequence object in order to avoid duplicate data when using multiprocessing.
* **steps**: Total number of steps (batches of samples) to yield from generator before stopping. Optional for Sequence: if unspecified, will use the len(generator) as a number of steps.
* **callbacks**: List of keras.callbacks.Callback instances. List of callbacks to apply during prediction. See [callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks).
* **max\_queue\_size**: Maximum size for the generator queue.
* **workers**: Integer. Maximum number of processes to spin up when using process-based threading. If unspecified, workers will default to 1. If 0, will execute the generator on the main thread.
* **use\_multiprocessing**: Boolean. If True, use process-based threading. If unspecified, use\_multiprocessing will default to False. Note that because this implementation relies on multiprocessing, you should not pass non-picklable **Arguments** to the generator as they can't be passed easily to children processes.
* **verbose**: verbosity mode, 0 or 1.

#### Returns:

Numpy array(s) of predictions.

#### Raises:

* **ValueError**: In case the generator yields data in an invalid format.

### predict\_on\_batch

predict\_on\_batch(x)

Returns predictions for a single batch of samples.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.

#### Returns:

Numpy array(s) of predictions.

#### Raises:

* **ValueError**: In case of mismatch between given number of inputs and expectations of the model.

### predict\_proba

predict\_proba(  
    x,  
    batch\_size=32,  
    verbose=0  
)

Generates class probability predictions for the input samples.

The input samples are processed batch by batch.

#### Arguments:

* **x**: input data, as a Numpy array or list of Numpy arrays (if the model has multiple inputs).
* **batch\_size**: integer.
* **verbose**: verbosity mode, 0 or 1.

#### Returns:

A Numpy array of probability predictions.

### reset\_metrics

reset\_metrics()

Resets the state of metrics.

### reset\_states

reset\_states()

### save

save(  
    filepath,  
    overwrite=True,  
    include\_optimizer=True,  
    save\_format=None  
)

Saves the model to Tensorflow SavedModel or a single HDF5 file.

#### The savefile includes:

* The model architecture, allowing to re-instantiate the model.
* The model weights.
* The state of the optimizer, allowing to resume training exactly where you left off.

This allows you to save the entirety of the state of a model in a single file.

Saved models can be reinstantiated via keras.models.load\_model. The model returned by load\_model is a compiled model ready to be used (unless the saved model was never compiled in the first place).

#### Arguments:

* **filepath**: String, path to SavedModel or H5 file to save the model.
* **overwrite**: Whether to silently overwrite any existing file at the target location, or provide the user with a manual prompt.
* **include\_optimizer**: If True, save optimizer's state together.
* **save\_format**: Either 'tf' or 'h5', indicating whether to save the model to Tensorflow SavedModel or HDF5. The default is currently 'h5', but will switch to 'tf' in TensorFlow 2.0. The 'tf' option is currently disabled (use [tf.keras.experimental.export\_saved\_model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model) instead).

#### Example:

from keras.models import load\_model  
  
model.save('my\_model.h5')  # creates a HDF5 file 'my\_model.h5'  
del model  # deletes the existing model  
  
# returns a compiled model  
# identical to the previous one  
model = load\_model('my\_model.h5')

### save\_weights

save\_weights(  
    filepath,  
    overwrite=True,  
    save\_format=None  
)

Saves all layer weights.

Either saves in HDF5 or in TensorFlow format based on the save\_format argument.

When saving in HDF5 format, the weight file has: - layer\_names (attribute), a list of strings (ordered names of model layers). - For every layer, a group named layer.name - For every such layer group, a group attribute weight\_names, a list of strings (ordered names of weights tensor of the layer). - For every weight in the layer, a dataset storing the weight value, named after the weight tensor.

When saving in TensorFlow format, all objects referenced by the network are saved in the same format as [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint), including any Layer instances or Optimizer instances assigned to object attributes. For networks constructed from inputs and outputs using tf.keras.Model(inputs, outputs), Layer instances used by the network are tracked/saved automatically. For user-defined classes which inherit from [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model), Layer instances must be assigned to object attributes, typically in the constructor. See the documentation of [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) and [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model)for details.

While the formats are the same, do not mix save\_weights and [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint). Checkpoints saved by Model.save\_weights should be loaded using Model.load\_weights. Checkpoints saved using [tf.train.Checkpoint.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#save) should be restored using the corresponding[tf.train.Checkpoint.restore](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#restore). Prefer [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) over save\_weights for training checkpoints.

The TensorFlow format matches objects and variables by starting at a root object, self for save\_weights, and greedily matching attribute names. For Model.save this is the Model, and for Checkpoint.save this is the Checkpoint even if the Checkpoint has a model attached. This means saving a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) using save\_weights and loading into a [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint)with a Model attached (or vice versa) will not match the Model's variables. See the [guide to training checkpoints](https://www.tensorflow.org/alpha/guide/checkpoints) for details on the TensorFlow format.

#### Arguments:

* **filepath**: String, path to the file to save the weights to. When saving in TensorFlow format, this is the prefix used for checkpoint files (multiple files are generated). Note that the '.h5' suffix causes weights to be saved in HDF5 format.
* **overwrite**: Whether to silently overwrite any existing file at the target location, or provide the user with a manual prompt.
* **save\_format**: Either 'tf' or 'h5'. A filepath ending in '.h5' or '.keras' will default to HDF5 if save\_format is None. Otherwise None defaults to 'tf'.

#### Raises:

* **ImportError**: If h5py is not available when attempting to save in HDF5 format.
* **ValueError**: For invalid/unknown format **Arguments**.

### summary

summary(  
    line\_length=None,  
    positions=None,  
    print\_fn=None  
)

Prints a string summary of the network.

#### Arguments:

* **line\_length**: Total length of printed lines (e.g. set this to adapt the display to different terminal window sizes).
* **positions**: Relative or absolute positions of log elements in each line. If not provided, defaults to [.33, .55, .67, 1.].
* **print\_fn**: Print function to use. Defaults to print. It will be called on each line of the summary. You can set it to a custom function in order to capture the string summary.

#### Raises:

* **ValueError**: if summary() is called before the model is built.

### test\_on\_batch

test\_on\_batch(  
    x,  
    y=None,  
    sample\_weight=None,  
    reset\_metrics=True  
)

Test the model on a single batch of samples.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset or a dataset iterator, y should not be specified (since targets will be obtained from the iterator).
* **sample\_weight**: Optional array of the same length as x, containing weights to apply to the model's loss for each sample. In the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specify sample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset or a dataset iterator.
* **reset\_metrics**: If True, the metrics returned will be only for this batch. If False, the metrics will be statefully accumulated across batches.

#### Returns:

Scalar test loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: In case of invalid user-provided **Arguments**.

### to\_json

to\_json(\*\*kwargs)

Returns a JSON string containing the network configuration.

To load a network from a JSON save file, use keras.models.model\_from\_json(json\_string, custom\_objects={}).

#### Arguments:

* **\*\*kwargs**: Additional keyword **Arguments** to be passed to json.dumps().

#### Returns:

A JSON string.

### to\_yaml

to\_yaml(\*\*kwargs)

Returns a yaml string containing the network configuration.

To load a network from a yaml save file, use keras.models.model\_from\_yaml(yaml\_string, custom\_objects={}).

custom\_objects should be a dictionary mapping the names of custom losses / layers / etc to the corresponding functions / classes.

#### Arguments:

* **\*\*kwargs**: Additional keyword **Arguments** to be passed to yaml.dump().

#### Returns:

A YAML string.

#### Raises:

* **ImportError**: if yaml module is not found.

### train\_on\_batch

train\_on\_batch(  
    x,  
    y=None,  
    sample\_weight=None,  
    class\_weight=None,  
    reset\_metrics=True  
)

Runs a single gradient update on a single batch of data.

#### Arguments:

* **x**: Input data. It could be:
  + A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
  + A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
  + A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
  + A [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) dataset or a dataset iterator.
* **y**: Target data. Like the input data x, it could be either Numpy array(s) or TensorFlow tensor(s). It should be consistent with x (you cannot have Numpy inputs and tensor targets, or inversely). If x is a dataset or a dataset iterator, y should not be specified (since targets will be obtained from the iterator).
* **sample\_weight**: Optional array of the same length as x, containing weights to apply to the model's loss for each sample. In the case of temporal data, you can pass a 2D array with shape (samples, sequence\_length), to apply a different weight to every timestep of every sample. In this case you should make sure to specify sample\_weight\_mode="temporal" in compile(). This argument is not supported when x is a dataset or a dataset iterator.
* **class\_weight**: Optional dictionary mapping class indices (integers) to a weight (float) to apply to the model's loss for the samples from this class during training. This can be useful to tell the model to "pay more attention" to samples from an under-represented class.
* **reset\_metrics**: If True, the metrics returned will be only for this batch. If False, the metrics will be statefully accumulated across batches.

#### Returns:

Scalar training loss (if the model has a single output and no metrics) or list of scalars (if the model has multiple outputs and/or metrics). The attribute model.metrics\_names will give you the display labels for the scalar outputs.

#### Raises:

* **ValueError**: In case of invalid user-provided **Arguments**.

Module: tf.keras.activations

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations#aliases)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations#functions)

Built-in activation functions.

Aliases:

* Module tf.compat.v2.keras.activations
* Module tf.keras.activations

Defined in [python/keras/api/\_v2/keras/activations/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/activations/__init__.py).

Functions

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/deserialize)

[elu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/elu): Exponential linear unit.

[exponential(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/exponential): Exponential activation function.

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/get)

[hard\_sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/hard_sigmoid): Hard sigmoid activation function.

[linear(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/linear): Linear activation function.

[relu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/relu): Rectified Linear Unit.

[selu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/selu): Scaled Exponential Linear Unit (SELU).

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/serialize)

tf.keras.activations.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/deserialize#aliases)

Aliases:

* tf.compat.v1.keras.activations.deserialize
* tf.compat.v2.keras.activations.deserialize
* tf.keras.activations.deserialize

tf.keras.activations.deserialize(  
    name,  
    custom\_objects=None  
)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

[sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/sigmoid): Sigmoid.

[softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softmax): The softmax activation function transforms the outputs so that all values are in

[softplus(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softplus): Softplus activation function.

[softsign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softsign): Softsign activation function.

[tanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/tanh): Hyperbolic Tangent activation function.

# tf.keras.activations.elu

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/elu#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/elu#aliases)

Exponential linear unit.

### Aliases:

* tf.compat.v1.keras.activations.elu
* tf.compat.v2.keras.activations.elu
* tf.keras.activations.elu

tf.keras.activations.elu(  
    x,  
    alpha=1.0  
)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

#### Arguments:

* **x**: Input tensor.
* **alpha**: A scalar, slope of negative section.

#### Returns:

The exponential linear activation: x if x > 0 and alpha \* (exp(x)-1) if x < 0.

#### Reference:

* [Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)](https://arxiv.org/abs/1511.07289)

# tf.keras.activations.exponential

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/exponential#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/exponential#aliases)

Exponential activation function.

### Aliases:

* tf.compat.v1.keras.activations.exponential
* tf.compat.v2.keras.activations.exponential
* tf.keras.activations.exponential

tf.keras.activations.exponential(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

#### Arguments:

* **x**: Input tensor.

#### Returns:

The exponential activation: exp(x).

tf.keras.activations.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/get#aliases)

Aliases:

* tf.compat.v1.keras.activations.get
* tf.compat.v2.keras.activations.get
* tf.keras.activations.get

tf.keras.activations.get(identifier)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

# tf.keras.activations.hard\_sigmoid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/hard_sigmoid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/hard_sigmoid#aliases)

Hard sigmoid activation function.

### Aliases:

* tf.compat.v1.keras.activations.hard\_sigmoid
* tf.compat.v2.keras.activations.hard\_sigmoid
* tf.keras.activations.hard\_sigmoid

tf.keras.activations.hard\_sigmoid(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

Faster to compute than sigmoid activation.

#### Arguments:

* **x**: Input tensor.

#### Returns:

Hard sigmoid activation: - 0 if x < -2.5 - 1 if x > 2.5 - 0.2 \* x + 0.5 if -2.5 <= x <= 2.5.

# tf.keras.activations.linear

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/linear#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/linear#aliases)

Linear activation function.

### Aliases:

* tf.compat.v1.keras.activations.linear
* tf.compat.v2.keras.activations.linear
* tf.keras.activations.linear

tf.keras.activations.linear(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

#### Arguments:

* **x**: Input tensor.

#### Returns:

The linear activation: x.

# tf.keras.activations.relu

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/relu#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/relu#aliases)

Rectified Linear Unit.

### Aliases:

* tf.compat.v1.keras.activations.relu
* tf.compat.v2.keras.activations.relu
* tf.keras.activations.relu

tf.keras.activations.relu(  
    x,  
    alpha=0.0,  
    max\_value=None,  
    threshold=0  
)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

With default values, it returns element-wise max(x, 0).

Otherwise, it follows: f(x) = max\_value for x >= max\_value, f(x) = x for threshold <= x < max\_value, f(x) = alpha \* (x - threshold) otherwise.

#### Arguments:

* **x**: A tensor or variable.
* **alpha**: A scalar, slope of negative section (default=0.).
* **max\_value**: float. Saturation threshold.
* **threshold**: float. Threshold value for thresholded activation.

#### Returns:

A tensor.

# tf.keras.activations.selu

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/selu#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/selu#aliases)

Scaled Exponential Linear Unit (SELU).

### Aliases:

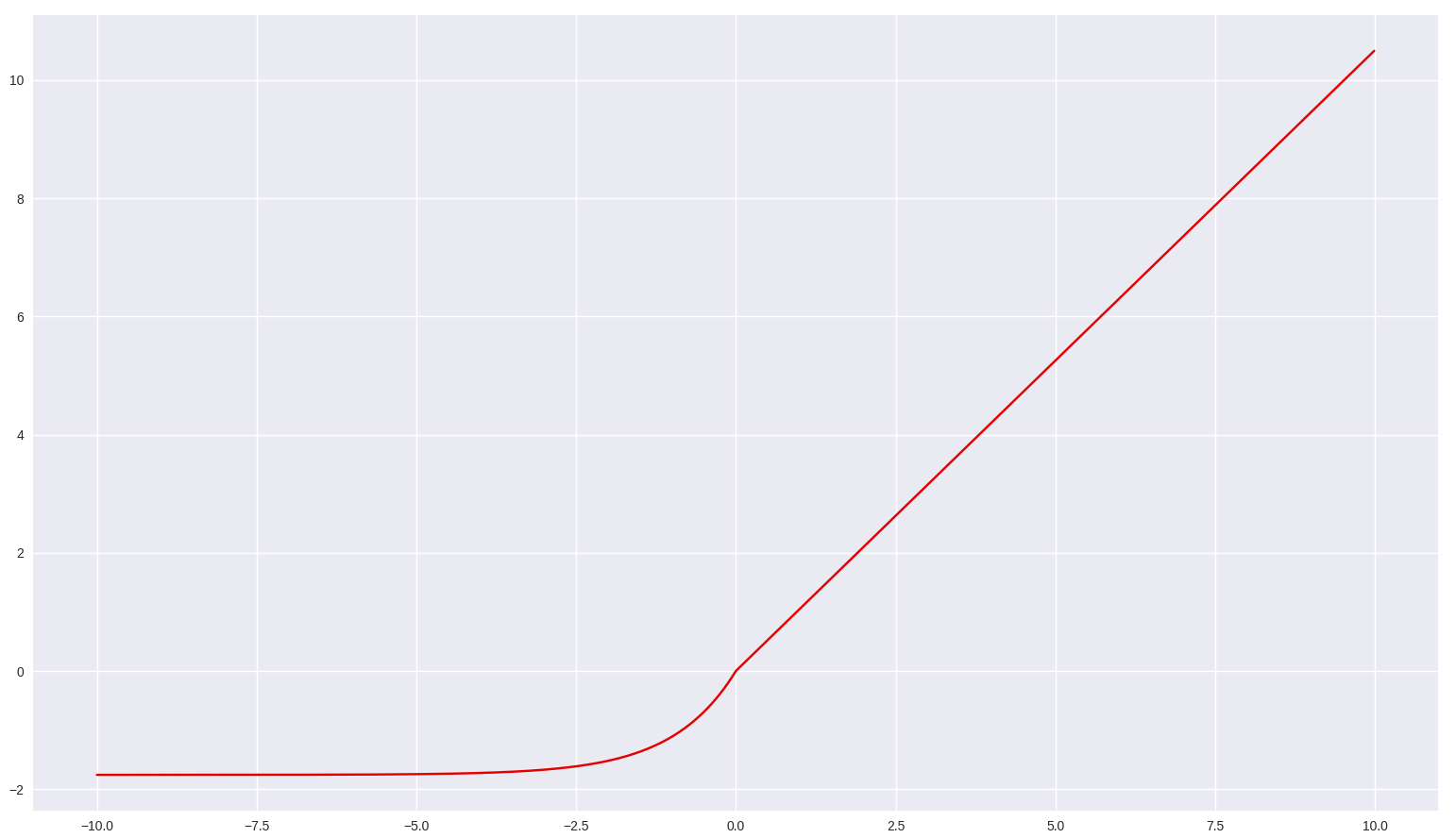
* tf.compat.v1.keras.activations.selu
* tf.compat.v2.keras.activations.selu
* tf.keras.activations.selu

tf.keras.activations.selu(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

The Scaled Exponential Linear Unit (SELU) activation function is: scale \* x if x > 0 and scale \* alpha \* (exp(x) - 1) if x < 0 where alpha and scale are pre-defined constants (alpha = 1.67326324 and scale = 1.05070098). The SELU activation function multiplies scale > 1 with the[elu](https://www.tensorflow.org/versions/r2.0/api\_docs/python/tf/keras/activations/elu)(Exponential Linear Unit (ELU)) to ensure a slope larger than one for positive net inputs.

The values of alpha and scale are chosen so that the mean and variance of the inputs are preserved between two consecutive layers as long as the weights are initialized correctly (see [lecun\_normal initialization](https://www.tensorflow.org/api_docs/python/tf/keras/initializers/lecun_normal) ) and the number of inputs is "large enough" (see references for more information).

 (Courtesy: Blog on Towards DataScience at https://towardsdatascience.com/selu-make-fnns-great-again-snn-8d61526802a9)

#### Example Usage:

n\_classes = 10 #10-class problem  
model = models.Sequential()  
model.add(Dense(64, kernel\_initializer='lecun\_normal', activation='selu',  
input\_shape=(28, 28, 1))))  
model.add(Dense(32, kernel\_initializer='lecun\_normal', activation='selu'))  
model.add(Dense(16, kernel\_initializer='lecun\_normal', activation='selu'))  
model.add(Dense(n\_classes, activation='softmax'))

#### Arguments:

* **x**: A tensor or variable to compute the activation function for.

#### Returns:

The scaled exponential unit activation: scale \* elu(x, alpha).

# Note

- To be used together with the initialization "[lecun\_normal]  
(https://www.tensorflow.org/api\_docs/python/tf/keras/initializers/lecun\_normal)".  
- To be used together with the dropout variant "[AlphaDropout]  
(https://www.tensorflow.org/api\_docs/python/tf/keras/layers/AlphaDropout)".

#### References:

[Self-Normalizing Neural Networks (Klambauer et al, 2017)](https://arxiv.org/abs/1706.02515)

tf.keras.activations.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/serialize#aliases)

Aliases:

* tf.compat.v1.keras.activations.serialize
* tf.compat.v2.keras.activations.serialize
* tf.keras.activations.serialize

tf.keras.activations.serialize(activation)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

# tf.keras.activations.sigmoid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/sigmoid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/sigmoid#aliases)

Sigmoid.

### Aliases:

* tf.compat.v1.keras.activations.sigmoid
* tf.compat.v2.keras.activations.sigmoid
* tf.keras.activations.sigmoid

tf.keras.activations.sigmoid(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

Applies the sigmoid activation function. The sigmoid function is defined as 1 divided by (1 + exp(-x)). It's curve is like an "S" and is like a smoothed version of the Heaviside (Unit Step Function) function. For small values (<-5) the sigmoid returns a value close to zero and for larger values (>5) the result of the function gets close to 1. **Arguments**: x: A tensor or variable.

#### Returns:

A tensor.

Sigmoid activation function.

#### Arguments:

* **x**: Input tensor.

#### Returns:

The sigmoid activation: (1.0 / (1.0 + exp(-x))).

# tf.keras.activations.softmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softmax#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softmax#aliases)

The softmax activation function transforms the outputs so that all values are in

### Aliases:

* tf.compat.v1.keras.activations.softmax
* tf.compat.v2.keras.activations.softmax
* tf.keras.activations.softmax

tf.keras.activations.softmax(  
    x,  
    axis=-1  
)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

range (0, 1) and sum to 1. It is often used as the activation for the last layer of a classification network because the result could be interpreted as a probability distribution. The softmax of x is calculated by exp(x)/tf.reduce\_sum(exp(x)).

#### Arguments:

* **x**: Input tensor.
* **axis**: Integer, axis along which the softmax normalization is applied.

#### Returns:

Tensor, output of softmax transformation (all values are non-negative and sum to 1).

#### Raises:

* **ValueError**: In case dim(x) == 1.

# tf.keras.activations.softplus

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softplus#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softplus#aliases)

Softplus activation function.

### Aliases:

* tf.compat.v1.keras.activations.softplus
* tf.compat.v2.keras.activations.softplus
* tf.keras.activations.softplus

tf.keras.activations.softplus(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

#### Arguments:

* **x**: Input tensor.

#### Returns:

The softplus activation: log(exp(x) + 1).

# tf.keras.activations.softsign

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softsign#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/softsign#aliases)

Softsign activation function.

### Aliases:

* tf.compat.v1.keras.activations.softsign
* tf.compat.v2.keras.activations.softsign
* tf.keras.activations.softsign

tf.keras.activations.softsign(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

#### Arguments:

* **x**: Input tensor.

#### Returns:

The softplus activation: x / (abs(x) + 1).

# tf.keras.activations.tanh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/tanh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/activations/tanh#aliases)

Hyperbolic Tangent activation function.

### Aliases:

* tf.compat.v1.keras.activations.tanh
* tf.compat.v2.keras.activations.tanh
* tf.keras.activations.tanh

tf.keras.activations.tanh(x)

Defined in [python/keras/activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/activations.py).

#### Arguments:

* **x**: Input tensor.

#### Returns:

The tanh activation: tanh(x) = sinh(x)/cosh(x) = ((exp(x) - exp(-x))/(exp(x) + exp(-x))).

Module: tf.keras.applications

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications#aliases)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications#modules)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications#functions)

Keras Applications are canned architectures with pre-trained weights.

Aliases:

* Module tf.compat.v2.keras.applications
* Module tf.keras.applications

Defined in [python/keras/api/\_v2/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/applications/__init__.py).

Modules

[densenet](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/densenet) module: DenseNet models for Keras.

[inception\_resnet\_v2](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_resnet_v2) module: Inception-ResNet V2 model for Keras.

[inception\_v3](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_v3) module: Inception V3 model for Keras.

[mobilenet](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet) module: MobileNet v1 models for Keras.

[mobilenet\_v2](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2) module: MobileNet v2 models for Keras.

[nasnet](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/nasnet) module: NASNet-A models for Keras.

[resnet50](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/resnet50) module: ResNet50 model for Keras.

[vgg16](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg16) module: VGG16 model for Keras.

[vgg19](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg19) module: VGG19 model for Keras.

[xception](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/xception) module: Xception V1 model for Keras.

Functions

[DenseNet121(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet121)

[DenseNet169(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet169)

[DenseNet201(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet201)

[InceptionResNetV2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionResNetV2)

[InceptionV3(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionV3)

[MobileNet(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNet)

[MobileNetV2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNetV2)

[NASNetLarge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/NASNetLarge)

[NASNetMobile(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/NASNetMobile)

[ResNet50(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/ResNet50)

[VGG16(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG16)

[VGG19(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG19)

[Xception(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/Xception)

tf.keras.applications.DenseNet121

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet121#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet121#aliases)

Aliases:

* tf.compat.v1.keras.applications.DenseNet121
* tf.compat.v1.keras.applications.densenet.DenseNet121
* tf.compat.v2.keras.applications.DenseNet121
* tf.compat.v2.keras.applications.densenet.DenseNet121
* tf.keras.applications.DenseNet121
* tf.keras.applications.densenet.DenseNet121

tf.keras.applications.DenseNet121(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.DenseNet169

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet169#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet169#aliases)

Aliases:

* tf.compat.v1.keras.applications.DenseNet169
* tf.compat.v1.keras.applications.densenet.DenseNet169
* tf.compat.v2.keras.applications.DenseNet169
* tf.compat.v2.keras.applications.densenet.DenseNet169
* tf.keras.applications.DenseNet169
* tf.keras.applications.densenet.DenseNet169

tf.keras.applications.DenseNet169(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.DenseNet201

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet201#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/DenseNet201#aliases)

Aliases:

* tf.compat.v1.keras.applications.DenseNet201
* tf.compat.v1.keras.applications.densenet.DenseNet201
* tf.compat.v2.keras.applications.DenseNet201
* tf.compat.v2.keras.applications.densenet.DenseNet201
* tf.keras.applications.DenseNet201
* tf.keras.applications.densenet.DenseNet201

tf.keras.applications.DenseNet201(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.InceptionResNetV2

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionResNetV2#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionResNetV2#aliases)

Aliases:

* tf.compat.v1.keras.applications.InceptionResNetV2
* tf.compat.v1.keras.applications.inception\_resnet\_v2.InceptionResNetV2
* tf.compat.v2.keras.applications.InceptionResNetV2
* tf.compat.v2.keras.applications.inception\_resnet\_v2.InceptionResNetV2
* tf.keras.applications.InceptionResNetV2
* tf.keras.applications.inception\_resnet\_v2.InceptionResNetV2

tf.keras.applications.InceptionResNetV2(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.InceptionV3

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionV3#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionV3#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/InceptionV3#used_in_the_tutorials)

Aliases:

* tf.compat.v1.keras.applications.InceptionV3
* tf.compat.v1.keras.applications.inception\_v3.InceptionV3
* tf.compat.v2.keras.applications.InceptionV3
* tf.compat.v2.keras.applications.inception\_v3.InceptionV3
* tf.keras.applications.InceptionV3
* tf.keras.applications.inception\_v3.InceptionV3

tf.keras.applications.InceptionV3(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)

tf.keras.applications.MobileNet

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNet#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNet#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNet#used_in_the_guide)

Aliases:

* tf.compat.v1.keras.applications.MobileNet
* tf.compat.v1.keras.applications.mobilenet.MobileNet
* tf.compat.v2.keras.applications.MobileNet
* tf.compat.v2.keras.applications.mobilenet.MobileNet
* tf.keras.applications.MobileNet
* tf.keras.applications.mobilenet.MobileNet

tf.keras.applications.MobileNet(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

tf.keras.applications.MobileNetV2

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNetV2#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNetV2#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/MobileNetV2#used_in_the_tutorials)

Aliases:

* tf.compat.v1.keras.applications.MobileNetV2
* tf.compat.v1.keras.applications.mobilenet\_v2.MobileNetV2
* tf.compat.v2.keras.applications.MobileNetV2
* tf.compat.v2.keras.applications.mobilenet\_v2.MobileNetV2
* tf.keras.applications.MobileNetV2
* tf.keras.applications.mobilenet\_v2.MobileNetV2

tf.keras.applications.MobileNetV2(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Transfer Learning Using Pretrained ConvNets](https://www.tensorflow.org/beta/tutorials/images/transfer_learning)

tf.keras.applications.NASNetLarge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/NASNetLarge#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/NASNetLarge#aliases)

Aliases:

* tf.compat.v1.keras.applications.NASNetLarge
* tf.compat.v1.keras.applications.nasnet.NASNetLarge
* tf.compat.v2.keras.applications.NASNetLarge
* tf.compat.v2.keras.applications.nasnet.NASNetLarge
* tf.keras.applications.NASNetLarge
* tf.keras.applications.nasnet.NASNetLarge

tf.keras.applications.NASNetLarge(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.NASNetMobile

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/NASNetMobile#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/NASNetMobile#aliases)

Aliases:

* tf.compat.v1.keras.applications.NASNetMobile
* tf.compat.v1.keras.applications.nasnet.NASNetMobile
* tf.compat.v2.keras.applications.NASNetMobile
* tf.compat.v2.keras.applications.nasnet.NASNetMobile
* tf.keras.applications.NASNetMobile
* tf.keras.applications.nasnet.NASNetMobile

tf.keras.applications.NASNetMobile(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.ResNet50

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/ResNet50#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/ResNet50#aliases)

Aliases:

* tf.compat.v1.keras.applications.ResNet50
* tf.compat.v1.keras.applications.resnet50.ResNet50
* tf.compat.v2.keras.applications.ResNet50
* tf.compat.v2.keras.applications.resnet50.ResNet50
* tf.keras.applications.ResNet50
* tf.keras.applications.resnet50.ResNet50

tf.keras.applications.ResNet50(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.VGG16

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG16#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG16#aliases)

Aliases:

* tf.compat.v1.keras.applications.VGG16
* tf.compat.v1.keras.applications.vgg16.VGG16
* tf.compat.v2.keras.applications.VGG16
* tf.compat.v2.keras.applications.vgg16.VGG16
* tf.keras.applications.VGG16
* tf.keras.applications.vgg16.VGG16

tf.keras.applications.VGG16(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.VGG19

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG19#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG19#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG19#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/VGG19#used_in_the_tutorials)

Aliases:

* tf.compat.v1.keras.applications.VGG19
* tf.compat.v1.keras.applications.vgg19.VGG19
* tf.compat.v2.keras.applications.VGG19
* tf.compat.v2.keras.applications.vgg19.VGG19
* tf.keras.applications.VGG19
* tf.keras.applications.vgg19.VGG19

tf.keras.applications.VGG19(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

tf.keras.applications.Xception

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/Xception#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/Xception#aliases)

Aliases:

* tf.compat.v1.keras.applications.Xception
* tf.compat.v1.keras.applications.xception.Xception
* tf.compat.v2.keras.applications.Xception
* tf.compat.v2.keras.applications.xception.Xception
* tf.keras.applications.Xception
* tf.keras.applications.xception.Xception

tf.keras.applications.Xception(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.densenet.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/densenet/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/densenet/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.densenet.decode\_predictions
* tf.compat.v2.keras.applications.densenet.decode\_predictions
* tf.keras.applications.densenet.decode\_predictions

tf.keras.applications.densenet.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.densenet.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/densenet/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/densenet/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.densenet.preprocess\_input
* tf.compat.v2.keras.applications.densenet.preprocess\_input
* tf.keras.applications.densenet.preprocess\_input

tf.keras.applications.densenet.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.inception\_resnet\_v2.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_resnet_v2/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_resnet_v2/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.inception\_resnet\_v2.decode\_predictions
* tf.compat.v2.keras.applications.inception\_resnet\_v2.decode\_predictions
* tf.keras.applications.inception\_resnet\_v2.decode\_predictions

tf.keras.applications.inception\_resnet\_v2.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.inception\_v3.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_v3/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_v3/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.inception\_v3.decode\_predictions
* tf.compat.v2.keras.applications.inception\_v3.decode\_predictions
* tf.keras.applications.inception\_v3.decode\_predictions

tf.keras.applications.inception\_v3.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.inception\_resnet\_v2.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_resnet_v2/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/inception_resnet_v2/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.inception\_resnet\_v2.preprocess\_input
* tf.compat.v2.keras.applications.inception\_resnet\_v2.preprocess\_input
* tf.keras.applications.inception\_resnet\_v2.preprocess\_input

tf.keras.applications.inception\_resnet\_v2.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.mobilenet.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.mobilenet.decode\_predictions
* tf.compat.v2.keras.applications.mobilenet.decode\_predictions
* tf.keras.applications.mobilenet.decode\_predictions

tf.keras.applications.mobilenet.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.mobilenet.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet/preprocess_input#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet/preprocess_input#used_in_the_guide)

Aliases:

* tf.compat.v1.keras.applications.mobilenet.preprocess\_input
* tf.compat.v2.keras.applications.mobilenet.preprocess\_input
* tf.keras.applications.mobilenet.preprocess\_input

tf.keras.applications.mobilenet.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

tf.keras.applications.mobilenet\_v2.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.mobilenet\_v2.decode\_predictions
* tf.compat.v2.keras.applications.mobilenet\_v2.decode\_predictions
* tf.keras.applications.mobilenet\_v2.decode\_predictions

tf.keras.applications.mobilenet\_v2.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.mobilenet\_v2.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.mobilenet\_v2.decode\_predictions
* tf.compat.v2.keras.applications.mobilenet\_v2.decode\_predictions
* tf.keras.applications.mobilenet\_v2.decode\_predictions

tf.keras.applications.mobilenet\_v2.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.mobilenet\_v2.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/mobilenet_v2/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.mobilenet\_v2.preprocess\_input
* tf.compat.v2.keras.applications.mobilenet\_v2.preprocess\_input
* tf.keras.applications.mobilenet\_v2.preprocess\_input

tf.keras.applications.mobilenet\_v2.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.nasnet.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/nasnet/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/nasnet/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.nasnet.decode\_predictions
* tf.compat.v2.keras.applications.nasnet.decode\_predictions
* tf.keras.applications.nasnet.decode\_predictions

tf.keras.applications.nasnet.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.nasnet.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/nasnet/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/nasnet/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.nasnet.preprocess\_input
* tf.compat.v2.keras.applications.nasnet.preprocess\_input
* tf.keras.applications.nasnet.preprocess\_input

tf.keras.applications.nasnet.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.resnet50.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/resnet50/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/resnet50/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.resnet50.decode\_predictions
* tf.compat.v2.keras.applications.resnet50.decode\_predictions
* tf.keras.applications.resnet50.decode\_predictions

tf.keras.applications.resnet50.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.resnet50.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/resnet50/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/resnet50/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.resnet50.preprocess\_input
* tf.compat.v2.keras.applications.resnet50.preprocess\_input
* tf.keras.applications.resnet50.preprocess\_input

tf.keras.applications.resnet50.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.vgg16.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg16/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg16/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.vgg16.decode\_predictions
* tf.compat.v2.keras.applications.vgg16.decode\_predictions
* tf.keras.applications.vgg16.decode\_predictions

tf.keras.applications.vgg16.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.vgg16.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg16/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg16/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.vgg16.preprocess\_input
* tf.compat.v2.keras.applications.vgg16.preprocess\_input
* tf.keras.applications.vgg16.preprocess\_input

tf.keras.applications.vgg16.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.vgg19.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg19/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg19/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.vgg19.decode\_predictions
* tf.compat.v2.keras.applications.vgg19.decode\_predictions
* tf.keras.applications.vgg19.decode\_predictions

tf.keras.applications.vgg19.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.vgg19.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg19/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg19/preprocess_input#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/vgg19/preprocess_input#used_in_the_tutorials)

Aliases:

* tf.compat.v1.keras.applications.vgg19.preprocess\_input
* tf.compat.v2.keras.applications.vgg19.preprocess\_input
* tf.keras.applications.vgg19.preprocess\_input

tf.keras.applications.vgg19.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

tf.keras.applications.xception.decode\_predictions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/xception/decode_predictions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/xception/decode_predictions#aliases)

Aliases:

* tf.compat.v1.keras.applications.xception.decode\_predictions
* tf.compat.v2.keras.applications.xception.decode\_predictions
* tf.keras.applications.xception.decode\_predictions

tf.keras.applications.xception.decode\_predictions(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

tf.keras.applications.xception.preprocess\_input

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/xception/preprocess_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/applications/xception/preprocess_input#aliases)

Aliases:

* tf.compat.v1.keras.applications.xception.preprocess\_input
* tf.compat.v2.keras.applications.xception.preprocess\_input
* tf.keras.applications.xception.preprocess\_input

tf.keras.applications.xception.preprocess\_input(  
    \*args,  
    \*\*kwargs  
)

Defined in [python/keras/applications/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/applications/__init__.py).

Module: tf.keras.backend

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend#aliases)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend#functions)

Keras backend API.

Aliases:

* Module tf.compat.v2.keras.backend
* Module tf.keras.backend

Defined in [python/keras/api/\_v2/keras/backend/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/backend/__init__.py).

Functions

[abs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/abs): Element-wise absolute value.

[all(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/all): Bitwise reduction (logical AND).

[any(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/any): Bitwise reduction (logical OR).

[arange(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/arange): Creates a 1D tensor containing a sequence of integers.

[argmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/argmax): Returns the index of the maximum value along an axis.

[argmin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/argmin): Returns the index of the minimum value along an axis.

[backend(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/backend): Publicly accessible method for determining the current backend.

[batch\_dot(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_dot): Batchwise dot product.

[batch\_flatten(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_flatten): Turn a nD tensor into a 2D tensor with same 0th dimension.

[batch\_get\_value(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_get_value): Returns the value of more than one tensor variable.

[batch\_normalization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_normalization): Applies batch normalization on x given mean, var, beta and gamma.

[batch\_set\_value(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_set_value): Sets the values of many tensor variables at once.

[bias\_add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/bias_add): Adds a bias vector to a tensor.

[binary\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/binary_crossentropy): Binary crossentropy between an output tensor and a target tensor.

[cast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cast): Casts a tensor to a different dtype and returns it.

[cast\_to\_floatx(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cast_to_floatx): Cast a Numpy array to the default Keras float type.

[categorical\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/categorical_crossentropy): Categorical crossentropy between an output tensor and a target tensor.

[clear\_session(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clear_session): Destroys the current TF graph and creates a new one.

[clip(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clip): Element-wise value clipping.

[concatenate(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/concatenate): Concatenates a list of tensors alongside the specified axis.

[constant(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/constant): Creates a constant tensor.

[conv1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv1d): 1D convolution.

[conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv2d): 2D convolution.

[conv2d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv2d_transpose): 2D deconvolution (i.e.

[conv3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv3d): 3D convolution.

[cos(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cos): Computes cos of x element-wise.

[count\_params(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/count_params): Returns the static number of elements in a variable or tensor.

[ctc\_batch\_cost(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_batch_cost): Runs CTC loss algorithm on each batch element.

[ctc\_decode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_decode): Decodes the output of a softmax.

[ctc\_label\_dense\_to\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_label_dense_to_sparse): Converts CTC labels from dense to sparse.

[cumprod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cumprod): Cumulative product of the values in a tensor, alongside the specified axis.

[cumsum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cumsum): Cumulative sum of the values in a tensor, alongside the specified axis.

[dot(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dot): Multiplies 2 tensors (and/or variables) and returns a *tensor*.

[dropout(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dropout): Sets entries in x to zero at random, while scaling the entire tensor.

[dtype(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dtype): Returns the dtype of a Keras tensor or variable, as a string.

[elu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/elu): Exponential linear unit.

[epsilon(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/epsilon): Returns the value of the fuzz factor used in numeric expressions.

[equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/equal): Element-wise equality between two tensors.

[eval(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/eval): Evaluates the value of a variable.

[exp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/exp): Element-wise exponential.

[expand\_dims(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/expand_dims): Adds a 1-sized dimension at index "axis".

[eye(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/eye): Instantiate an identity matrix and returns it.

[flatten(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/flatten): Flatten a tensor.

[floatx(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/floatx): Returns the default float type, as a string.

[foldl(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/foldl): Reduce elems using fn to combine them from left to right.

[foldr(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/foldr): Reduce elems using fn to combine them from right to left.

[function(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/function): Instantiates a Keras function.

[gather(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/gather): Retrieves the elements of indices indices in the tensor reference.

[get\_uid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_uid): Associates a string prefix with an integer counter in a TensorFlow graph.

[get\_value(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_value): Returns the value of a variable.

[gradients(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/gradients): Returns the gradients of loss w.r.t. variables.

[greater(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/greater): Element-wise truth value of (x > y).

[greater\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/greater_equal): Element-wise truth value of (x >= y).

[hard\_sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/hard_sigmoid): Segment-wise linear approximation of sigmoid.

[image\_data\_format(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/image_data_format): Returns the default image data format convention.

[in\_test\_phase(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_test_phase): Selects x in test phase, and alt otherwise.

[in\_top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_top_k): Returns whether the targets are in the top k predictions.

[in\_train\_phase(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_train_phase): Selects x in train phase, and alt otherwise.

[int\_shape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/int_shape): Returns the shape of tensor or variable as a tuple of int or None entries.

[is\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/is_sparse): Returns whether a tensor is a sparse tensor.

[l2\_normalize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/l2_normalize): Normalizes a tensor wrt the L2 norm alongside the specified axis.

[learning\_phase(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/learning_phase): Returns the learning phase flag.

[learning\_phase\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/learning_phase_scope): Provides a scope within which the learning phase is equal to value.

[less(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/less): Element-wise truth value of (x < y).

[less\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/less_equal): Element-wise truth value of (x <= y).

[local\_conv1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/local_conv1d): Apply 1D conv with un-shared weights.

[local\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/local_conv2d): Apply 2D conv with un-shared weights.

[log(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/log): Element-wise log.

[manual\_variable\_initialization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/manual_variable_initialization): Sets the manual variable initialization flag.

[map\_fn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/map_fn): Map the function fn over the elements elems and return the outputs.

[max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/max): Maximum value in a tensor.

[maximum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/maximum): Element-wise maximum of two tensors.

[mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/mean): Mean of a tensor, alongside the specified axis.

[min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/min): Minimum value in a tensor.

[minimum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/minimum): Element-wise minimum of two tensors.

[moving\_average\_update(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/moving_average_update): Compute the moving average of a variable.

[name\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/name_scope): A context manager for use when defining a Python op.

[ndim(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ndim): Returns the number of axes in a tensor, as an integer.

[normalize\_batch\_in\_training(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/normalize_batch_in_training): Computes mean and std for batch then apply batch\_normalization on batch.

[not\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/not_equal): Element-wise inequality between two tensors.

[one\_hot(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/one_hot): Computes the one-hot representation of an integer tensor.

[ones(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ones): Instantiates an all-ones variable and returns it.

[ones\_like(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ones_like): Instantiates an all-ones variable of the same shape as another tensor.

[permute\_dimensions(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/permute_dimensions): Permutes axes in a tensor.

[placeholder(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/placeholder): Instantiates a placeholder tensor and returns it.

[pool2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pool2d): 2D Pooling.

[pool3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pool3d): 3D Pooling.

[pow(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pow): Element-wise exponentiation.

[print\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/print_tensor): Prints message and the tensor value when evaluated.

[prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/prod): Multiplies the values in a tensor, alongside the specified axis.

[random\_binomial(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_binomial): Returns a tensor with random binomial distribution of values.

[random\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal): Returns a tensor with normal distribution of values.

[random\_normal\_variable(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal_variable): Instantiates a variable with values drawn from a normal distribution.

[random\_uniform(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_uniform): Returns a tensor with uniform distribution of values.

[random\_uniform\_variable(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_uniform_variable): Instantiates a variable with values drawn from a uniform distribution.

[relu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/relu): Rectified linear unit.

[repeat(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/repeat): Repeats a 2D tensor.

[repeat\_elements(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/repeat_elements): Repeats the elements of a tensor along an axis, like np.repeat.

[reset\_uids(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reset_uids): Resets graph identifiers.

[reshape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reshape): Reshapes a tensor to the specified shape.

[resize\_images(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/resize_images): Resizes the images contained in a 4D tensor.

[resize\_volumes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/resize_volumes): Resizes the volume contained in a 5D tensor.

[reverse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reverse): Reverse a tensor along the specified axes.

[rnn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/rnn): Iterates over the time dimension of a tensor.

[round(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/round): Element-wise rounding to the closest integer.

[separable\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/separable_conv2d): 2D convolution with separable filters.

[set\_epsilon(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_epsilon): Sets the value of the fuzz factor used in numeric expressions.

[set\_floatx(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_floatx): Sets the default float type.

[set\_image\_data\_format(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_image_data_format): Sets the value of the image data format convention.

[set\_learning\_phase(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_learning_phase): Sets the learning phase to a fixed value.

[set\_value(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_value): Sets the value of a variable, from a Numpy array.

[shape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/shape): Returns the symbolic shape of a tensor or variable.

[sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sigmoid): Element-wise sigmoid.

[sign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sign): Element-wise sign.

[sin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sin): Computes sin of x element-wise.

[softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softmax): Softmax of a tensor.

[softplus(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softplus): Softplus of a tensor.

[softsign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softsign): Softsign of a tensor.

[sparse\_categorical\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sparse_categorical_crossentropy): Categorical crossentropy with integer targets.

[spatial\_2d\_padding(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/spatial_2d_padding): Pads the 2nd and 3rd dimensions of a 4D tensor.

[spatial\_3d\_padding(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/spatial_3d_padding): Pads 5D tensor with zeros along the depth, height, width dimensions.

[sqrt(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sqrt): Element-wise square root.

[square(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/square): Element-wise square.

[squeeze(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/squeeze): Removes a 1-dimension from the tensor at index "axis".

[stack(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/stack): Stacks a list of rank R tensors into a rank R+1 tensor.

[std(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/std): Standard deviation of a tensor, alongside the specified axis.

[stop\_gradient(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/stop_gradient): Returns variables but with zero gradient w.r.t. every other variable.

[sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sum): Sum of the values in a tensor, alongside the specified axis.

[switch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/switch): Switches between two operations depending on a scalar value.

[tanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/tanh): Element-wise tanh.

[temporal\_padding(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/temporal_padding): Pads the middle dimension of a 3D tensor.

[tile(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/tile): Creates a tensor by tiling x by n.

[to\_dense(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/to_dense): Converts a sparse tensor into a dense tensor and returns it.

[transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/transpose): Transposes a tensor and returns it.

[truncated\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/truncated_normal): Returns a tensor with truncated random normal distribution of values.

[update(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update)

[update\_add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update_add): Update the value of x by adding increment.

[update\_sub(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update_sub): Update the value of x by subtracting decrement.

[var(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/var): Variance of a tensor, alongside the specified axis.

[variable(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/variable): Instantiates a variable and returns it.

[zeros(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/zeros): Instantiates an all-zeros variable and returns it.

[zeros\_like(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/zeros_like): Instantiates an all-zeros variable of the same shape as another tensor.

tf.compat.v1.keras.backend.get\_session

Returns the TF session to be used by the backend.

tf.compat.v1.keras.backend.get\_session(op\_input\_list=())

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

If a default TensorFlow session is available, we will return it.

Else, we will return the global Keras session assuming it matches the current graph.

If no global Keras session exists at this point: we will create a new global session.

Note that you can manually set the global session via K.set\_session(sess).

**Arguments**:

* **op\_input\_list**: An option sequence of tensors or ops, which will be used to determine the current graph. Otherwise the default graph will be used.

Returns:

A TensorFlow session.

tf.compat.v1.keras.backend.set\_session

Sets the global TensorFlow session.

tf.compat.v1.keras.backend.set\_session(session)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

**Arguments**:

* **session**: A TF Session.

# tf.keras.backend.abs

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/abs#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/abs#aliases)

Element-wise absolute value.

### Aliases:

* tf.compat.v1.keras.backend.abs
* tf.compat.v2.keras.backend.abs
* tf.keras.backend.abs

tf.keras.backend.abs(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.all

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/all#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/all#aliases)

Bitwise reduction (logical AND).

### Aliases:

* tf.compat.v1.keras.backend.all
* tf.compat.v2.keras.backend.all
* tf.keras.backend.all

tf.keras.backend.all(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **axis**: axis along which to perform the reduction.
* **keepdims**: whether the drop or broadcast the reduction axes.

#### Returns:

A uint8 tensor (0s and 1s).

# tf.keras.backend.any

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/any#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/any#aliases)

Bitwise reduction (logical OR).

### Aliases:

* tf.compat.v1.keras.backend.any
* tf.compat.v2.keras.backend.any
* tf.keras.backend.any

tf.keras.backend.any(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **axis**: axis along which to perform the reduction.
* **keepdims**: whether the drop or broadcast the reduction axes.

#### Returns:

A uint8 tensor (0s and 1s).

# tf.keras.backend.arange

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/arange#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/arange#aliases)

Creates a 1D tensor containing a sequence of integers.

### Aliases:

* tf.compat.v1.keras.backend.arange
* tf.compat.v2.keras.backend.arange
* tf.keras.backend.arange

tf.keras.backend.arange(  
    start,  
    stop=None,  
    step=1,  
    dtype='int32'  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

The function **Arguments** use the same convention as Theano's arange: if only one argument is provided, it is in fact the "stop" argument and "start" is 0.

The default type of the returned tensor is 'int32' to match TensorFlow's default.

#### Arguments:

* **start**: Start value.
* **stop**: Stop value.
* **step**: Difference between two successive values.
* **dtype**: Integer dtype to use.

#### Returns:

An integer tensor.

# tf.keras.backend.argmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/argmax#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/argmax#aliases)

Returns the index of the maximum value along an axis.

### Aliases:

* tf.compat.v1.keras.backend.argmax
* tf.compat.v2.keras.backend.argmax
* tf.keras.backend.argmax

tf.keras.backend.argmax(  
    x,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **axis**: axis along which to perform the reduction.

#### Returns:

# A tensor.

# tf.keras.backend.argmin

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/argmin#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/argmin#aliases)

Returns the index of the minimum value along an axis.

### Aliases:

* tf.compat.v1.keras.backend.argmin
* tf.compat.v2.keras.backend.argmin
* tf.keras.backend.argmin

tf.keras.backend.argmin(  
    x,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **axis**: axis along which to perform the reduction.

#### Returns:

A tensor.

# tf.keras.backend.backend

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/backend#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/backend#aliases)

Publicly accessible method for determining the current backend.

### Aliases:

* tf.compat.v1.keras.backend.backend
* tf.compat.v2.keras.backend.backend
* tf.keras.backend.backend

tf.keras.backend.backend()

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Only exists for API compatibility with multi-backend Keras.

#### Returns:

The string "tensorflow".

# tf.keras.backend.batch\_dot

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_dot#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_dot#aliases)

Batchwise dot product.

### Aliases:

* tf.compat.v1.keras.backend.batch\_dot
* tf.compat.v2.keras.backend.batch\_dot
* tf.keras.backend.batch\_dot

tf.keras.backend.batch\_dot(  
    x,  
    y,  
    axes=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

batch\_dot is used to compute dot product of x and y when x and y are data in batch, i.e. in a shape of (batch\_size, :). batch\_dot results in a tensor or variable with less dimensions than the input. If the number of dimensions is reduced to 1, we use expand\_dims to make sure that ndim is at least 2.

#### Arguments:

* **x**: Keras tensor or variable with ndim >= 2.
* **y**: Keras tensor or variable with ndim >= 2.
* **axes**: list of (or single) int with target dimensions. The lengths of axes[0] and axes[1]should be the same.

#### Returns:

A tensor with shape equal to the concatenation of x's shape (less the dimension that was summed over) and y's shape (less the batch dimension and the dimension that was summed over). If the final rank is 1, we reshape it to (batch\_size, 1).

#### Examples:

Assume x = [[1, 2], [3, 4]] and y = [[5, 6], [7, 8]] batch\_dot(x, y, axes=1) = [[17, 53]] which is the main diagonal of x.dot(y.T), although we never have to calculate the off-diagonal elements.

Shape inference: Let x's shape be (100, 20) and y's shape be (100, 30, 20). If axes is (1, 2), to find the output shape of resultant tensor, loop through each dimension in x's shape and y's shape:

* x.shape[0] : 100 : append to output shape
* x.shape[1] : 20 : do not append to output shape, dimension 1 of x has been summed over. (dot\_axes[0] = 1)
* y.shape[0] : 100 : do not append to output shape, always ignore first dimension of y
* y.shape[1] : 30 : append to output shape
* y.shape[2] : 20 : do not append to output shape, dimension 2 of y has been summed over. (dot\_axes[1] = 2) output\_shape = (100, 30)

    >>> x\_batch = K.ones(shape=(32, 20, 1))  
    >>> y\_batch = K.ones(shape=(32, 30, 20))  
    >>> xy\_batch\_dot = K.batch\_dot(x\_batch, y\_batch, axes=[1, 2])  
    >>> K.int\_shape(xy\_batch\_dot)  
    (32, 1, 30)

# tf.keras.backend.batch\_flatten

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_flatten#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_flatten#aliases)

Turn a nD tensor into a 2D tensor with same 0th dimension.

### Aliases:

* tf.compat.v1.keras.backend.batch\_flatten
* tf.compat.v2.keras.backend.batch\_flatten
* tf.keras.backend.batch\_flatten

tf.keras.backend.batch\_flatten(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

In other words, it flattens each data samples of a batch.

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor.

#### Examples:

Flattening a 3D tensor to 2D by collapsing the last dimension.

    >>> from tensorflow.keras import backend as K  
    >>> x\_batch = K.ones(shape=(2, 3, 4, 5))  
    >>> x\_batch\_flatten = K.batch\_flatten(x\_batch)  
    >>> K.int\_shape(x\_batch\_flatten)  
    (2, 60)

# tf.keras.backend.batch\_get\_value

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_get_value#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_get_value#aliases)

Returns the value of more than one tensor variable.

### Aliases:

* tf.compat.v1.keras.backend.batch\_get\_value
* tf.compat.v2.keras.backend.batch\_get\_value
* tf.keras.backend.batch\_get\_value

tf.keras.backend.batch\_get\_value(tensors)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **tensors**: list of ops to run.

#### Returns:

A list of Numpy arrays.

#### Raises:

* **RuntimeError**: If this method is called inside defun.

# tf.keras.backend.batch\_normalization

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_normalization#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_normalization#aliases)

Applies batch normalization on x given mean, var, beta and gamma.

### Aliases:

* tf.compat.v1.keras.backend.batch\_normalization
* tf.compat.v2.keras.backend.batch\_normalization
* tf.keras.backend.batch\_normalization

tf.keras.backend.batch\_normalization(  
    x,  
    mean,  
    var,  
    beta,  
    gamma,  
    axis=-1,  
    epsilon=0.001  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

I.e. returns: output = (x - mean) / (sqrt(var) + epsilon) \* gamma + beta

#### Arguments:

* **x**: Input tensor or variable.
* **mean**: Mean of batch.
* **var**: Variance of batch.
* **beta**: Tensor with which to center the input.
* **gamma**: Tensor by which to scale the input.
* **axis**: Integer, the axis that should be normalized. (typically the features axis).
* **epsilon**: Fuzz factor.

#### Returns:

A tensor.

# tf.keras.backend.batch\_set\_value

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_set_value#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/batch_set_value#aliases)

Sets the values of many tensor variables at once.

### Aliases:

* tf.compat.v1.keras.backend.batch\_set\_value
* tf.compat.v2.keras.backend.batch\_set\_value
* tf.keras.backend.batch\_set\_value

tf.keras.backend.batch\_set\_value(tuples)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **tuples**: a list of tuples (tensor, value). value should be a Numpy array.

# tf.keras.backend.bias\_add

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/bias_add#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/bias_add#aliases)

Adds a bias vector to a tensor.

### Aliases:

* tf.compat.v1.keras.backend.bias\_add
* tf.compat.v2.keras.backend.bias\_add
* tf.keras.backend.bias\_add

tf.keras.backend.bias\_add(  
    x,  
    bias,  
    data\_format=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **bias**: Bias tensor to add.
* **data\_format**: string, "channels\_last" or "channels\_first".

#### Returns:

Output tensor.

#### Raises:

* **ValueError**: In one of the two cases below: 1. invalid data\_format argument. 2. invalid bias shape. the bias should be either a vector or a tensor with ndim(x) - 1 dimension

# tf.keras.backend.binary\_crossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/binary_crossentropy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/binary_crossentropy#aliases)

Binary crossentropy between an output tensor and a target tensor.

### Aliases:

* tf.compat.v1.keras.backend.binary\_crossentropy
* tf.compat.v2.keras.backend.binary\_crossentropy
* tf.keras.backend.binary\_crossentropy

tf.keras.backend.binary\_crossentropy(  
    target,  
    output,  
    from\_logits=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **target**: A tensor with the same shape as output.
* **output**: A tensor.
* **from\_logits**: Whether output is expected to be a logits tensor. By default, we consider that output encodes a probability distribution.

#### Returns:

A tensor.

# tf.keras.backend.cast

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cast#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cast#aliases)

Casts a tensor to a different dtype and returns it.

### Aliases:

* tf.compat.v1.keras.backend.cast
* tf.compat.v2.keras.backend.cast
* tf.keras.backend.cast

tf.keras.backend.cast(  
    x,  
    dtype  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

You can cast a Keras variable but it still returns a Keras tensor.

#### Arguments:

* **x**: Keras tensor (or variable).
* **dtype**: String, either ('float16', 'float32', or 'float64').

#### Returns:

Keras tensor with dtype dtype.

#### Examples:

Cast a float32 variable to a float64 tensor

    >>> import tensorflow as tf  
    >>> from tensorflow.keras import backend as K  
    >>> input = K.ones(shape=(1,3))  
    >>> print(input)  
    >>> cast\_input = K.cast(input, dtype='float64')  
    >>> print(cast\_input)  
  
    <tf.Variable 'Variable:0' shape=(1, 3) dtype=float32,  
         numpy=array([[1., 1., 1.]], dtype=float32)>  
    tf.Tensor([[1. 1. 1.]], shape=(1, 3), dtype=float64)

# tf.keras.backend.cast\_to\_floatx

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cast_to_floatx#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cast_to_floatx#aliases)

Cast a Numpy array to the default Keras float type.

### Aliases:

* tf.compat.v1.keras.backend.cast\_to\_floatx
* tf.compat.v2.keras.backend.cast\_to\_floatx
* tf.keras.backend.cast\_to\_floatx

tf.keras.backend.cast\_to\_floatx(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Numpy array.

#### Returns:

The same Numpy array, cast to its new type.

#### Example:

    >>> from keras import backend as K  
    >>> K.floatx()  
    'float32'  
    >>> arr = numpy.array([1.0, 2.0], dtype='float64')  
    >>> arr.dtype  
    dtype('float64')  
    >>> new\_arr = K.cast\_to\_floatx(arr)  
    >>> new\_arr  
    array([ 1.,  2.], dtype=float32)  
    >>> new\_arr.dtype  
    dtype('float32')

# tf.keras.backend.categorical\_crossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/categorical_crossentropy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/categorical_crossentropy#aliases)

Categorical crossentropy between an output tensor and a target tensor.

### Aliases:

* tf.compat.v1.keras.backend.categorical\_crossentropy
* tf.compat.v2.keras.backend.categorical\_crossentropy
* tf.keras.backend.categorical\_crossentropy

tf.keras.backend.categorical\_crossentropy(  
    target,  
    output,  
    from\_logits=False,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **target**: A tensor of the same shape as output.
* **output**: A tensor resulting from a softmax (unless from\_logits is True, in which case output is expected to be the logits).
* **from\_logits**: Boolean, whether output is the result of a softmax, or is a tensor of logits.
* **axis**: Int specifying the channels axis. axis=-1 corresponds to data format channels\_last', andaxis=1corresponds to data formatchannels\_first`.

#### Returns:

Output tensor.

#### Raises:

* **ValueError**: if axis is neither -1 nor one of the axes of output.

tf.keras.backend.clear\_session

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clear_session#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clear_session#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clear_session#used_in_the_guide)

Destroys the current TF graph and creates a new one.

Aliases:

* tf.compat.v1.keras.backend.clear\_session
* tf.compat.v2.keras.backend.clear\_session
* tf.keras.backend.clear\_session

tf.keras.backend.clear\_session()

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Used in the guide:

* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

Useful to avoid clutter from old models / layers.

# tf.keras.backend.clip

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clip#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/clip#aliases)

Element-wise value clipping.

### Aliases:

* tf.compat.v1.keras.backend.clip
* tf.compat.v2.keras.backend.clip
* tf.keras.backend.clip

tf.keras.backend.clip(  
    x,  
    min\_value,  
    max\_value  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **min\_value**: Python float or integer.
* **max\_value**: Python float or integer.

#### Returns:

A tensor.

# tf.keras.backend.concatenate

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/concatenate#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/concatenate#aliases)

Concatenates a list of tensors alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.concatenate
* tf.compat.v2.keras.backend.concatenate
* tf.keras.backend.concatenate

tf.keras.backend.concatenate(  
    tensors,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **tensors**: list of tensors to concatenate.
* **axis**: concatenation axis.

#### Returns:

A tensor.

# tf.keras.backend.constant

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/constant#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/constant#aliases)

Creates a constant tensor.

### Aliases:

* tf.compat.v1.keras.backend.constant
* tf.compat.v2.keras.backend.constant
* tf.keras.backend.constant

tf.keras.backend.constant(  
    value,  
    dtype=None,  
    shape=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **value**: A constant value (or list)
* **dtype**: The type of the elements of the resulting tensor.
* **shape**: Optional dimensions of resulting tensor.
* **name**: Optional name for the tensor.

#### Returns:

A Constant Tensor.

# tf.keras.backend.conv1d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv1d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv1d#aliases)

1D convolution.

### Aliases:

* tf.compat.v1.keras.backend.conv1d
* tf.compat.v2.keras.backend.conv1d
* tf.keras.backend.conv1d

tf.keras.backend.conv1d(  
    x,  
    kernel,  
    strides=1,  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **kernel**: kernel tensor.
* **strides**: stride integer.
* **padding**: string, "same", "causal" or "valid".
* **data\_format**: string, one of "channels\_last", "channels\_first".
* **dilation\_rate**: integer dilate rate.

#### Returns:

A tensor, result of 1D convolution.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.conv2d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv2d#aliases)

2D convolution.

### Aliases:

* tf.compat.v1.keras.backend.conv2d
* tf.compat.v2.keras.backend.conv2d
* tf.keras.backend.conv2d

tf.keras.backend.conv2d(  
    x,  
    kernel,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1)  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **kernel**: kernel tensor.
* **strides**: strides tuple.
* **padding**: string, "same" or "valid".
* **data\_format**: "channels\_last" or "channels\_first".
* **dilation\_rate**: tuple of 2 integers.

#### Returns:

A tensor, result of 2D convolution.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.conv2d\_transpose

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv2d_transpose#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv2d_transpose#aliases)

2D deconvolution (i.e.

### Aliases:

* tf.compat.v1.keras.backend.conv2d\_transpose
* tf.compat.v2.keras.backend.conv2d\_transpose
* tf.keras.backend.conv2d\_transpose

tf.keras.backend.conv2d\_transpose(  
    x,  
    kernel,  
    output\_shape,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1)  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

transposed convolution).

#### Arguments:

* **x**: Tensor or variable.
* **kernel**: kernel tensor.
* **output\_shape**: 1D int tensor for the output shape.
* **strides**: strides tuple.
* **padding**: string, "same" or "valid".
* **data\_format**: string, "channels\_last" or "channels\_first".
* **dilation\_rate**: Tuple of 2 integers.

#### Returns:

A tensor, result of transposed 2D convolution.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.conv3d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/conv3d#aliases)

3D convolution.

### Aliases:

* tf.compat.v1.keras.backend.conv3d
* tf.compat.v2.keras.backend.conv3d
* tf.keras.backend.conv3d

tf.keras.backend.conv3d(  
    x,  
    kernel,  
    strides=(1, 1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1, 1)  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **kernel**: kernel tensor.
* **strides**: strides tuple.
* **padding**: string, "same" or "valid".
* **data\_format**: string, "channels\_last" or "channels\_first".
* **dilation\_rate**: tuple of 3 integers.

#### Returns:

A tensor, result of 3D convolution.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.cos

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cos#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cos#aliases)

Computes cos of x element-wise.

### Aliases:

* tf.compat.v1.keras.backend.cos
* tf.compat.v2.keras.backend.cos
* tf.keras.backend.cos

tf.keras.backend.cos(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.count\_params

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/count_params#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/count_params#aliases)

Returns the static number of elements in a variable or tensor.

### Aliases:

* tf.compat.v1.keras.backend.count\_params
* tf.compat.v2.keras.backend.count\_params
* tf.keras.backend.count\_params

tf.keras.backend.count\_params(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Variable or tensor.

#### Returns:

Integer, the number of scalars in x.

#### Example:

    >>> kvar = K.zeros((2,3))  
    >>> K.count\_params(kvar)  
    6  
    >>> K.eval(kvar)  
    array([[ 0.,  0.,  0.],  
           [ 0.,  0.,  0.]], dtype=float32)

# tf.keras.backend.ctc\_batch\_cost

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_batch_cost#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_batch_cost#aliases)

Runs CTC loss algorithm on each batch element.

### Aliases:

* tf.compat.v1.keras.backend.ctc\_batch\_cost
* tf.compat.v2.keras.backend.ctc\_batch\_cost
* tf.keras.backend.ctc\_batch\_cost

tf.keras.backend.ctc\_batch\_cost(  
    y\_true,  
    y\_pred,  
    input\_length,  
    label\_length  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **y\_true**: tensor (samples, max\_string\_length) containing the truth labels.
* **y\_pred**: tensor (samples, time\_steps, num\_categories) containing the prediction, or output of the softmax.
* **input\_length**: tensor (samples, 1) containing the sequence length for each batch item in y\_pred.
* **label\_length**: tensor (samples, 1) containing the sequence length for each batch item in y\_true.

#### Returns:

Tensor with shape (samples,1) containing the CTC loss of each element.

# tf.keras.backend.ctc\_decode

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_decode#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_decode#aliases)

Decodes the output of a softmax.

### Aliases:

* tf.compat.v1.keras.backend.ctc\_decode
* tf.compat.v2.keras.backend.ctc\_decode
* tf.keras.backend.ctc\_decode

tf.keras.backend.ctc\_decode(  
    y\_pred,  
    input\_length,  
    greedy=True,  
    beam\_width=100,  
    top\_paths=1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Can use either greedy search (also known as best path) or a constrained dictionary search.

#### Arguments:

* **y\_pred**: tensor (samples, time\_steps, num\_categories) containing the prediction, or output of the softmax.
* **input\_length**: tensor (samples, ) containing the sequence length for each batch item in y\_pred.
* **greedy**: perform much faster best-path search if true. This does not use a dictionary.
* **beam\_width**: if greedy is false: a beam search decoder will be used with a beam of this width.
* **top\_paths**: if greedy is false, how many of the most probable paths will be returned.

#### Returns:

* **Tuple**: List: if greedy is true, returns a list of one element that contains the decoded sequence. If false, returns the top\_paths most probable decoded sequences. Important: blank labels are returned as -1. Tensor (top\_paths, ) that contains the log probability of each decoded sequence.

# tf.keras.backend.ctc\_label\_dense\_to\_sparse

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_label_dense_to_sparse#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ctc_label_dense_to_sparse#aliases)

Converts CTC labels from dense to sparse.

### Aliases:

* tf.compat.v1.keras.backend.ctc\_label\_dense\_to\_sparse
* tf.compat.v2.keras.backend.ctc\_label\_dense\_to\_sparse
* tf.keras.backend.ctc\_label\_dense\_to\_sparse

tf.keras.backend.ctc\_label\_dense\_to\_sparse(  
    labels,  
    label\_lengths  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **labels**: dense CTC labels.
* **label\_lengths**: length of the labels.

#### Returns:

A sparse tensor representation of the labels.

# tf.keras.backend.cumprod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cumprod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cumprod#aliases)

Cumulative product of the values in a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.cumprod
* tf.compat.v2.keras.backend.cumprod
* tf.keras.backend.cumprod

tf.keras.backend.cumprod(  
    x,  
    axis=0  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to compute the product.

#### Returns:

A tensor of the cumulative product of values of x along axis.

# tf.keras.backend.cumsum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cumsum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/cumsum#aliases)

Cumulative sum of the values in a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.cumsum
* tf.compat.v2.keras.backend.cumsum
* tf.keras.backend.cumsum

tf.keras.backend.cumsum(  
    x,  
    axis=0  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to compute the sum.

#### Returns:

A tensor of the cumulative sum of values of x along axis.

# tf.keras.backend.dot

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dot#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dot#aliases)

Multiplies 2 tensors (and/or variables) and returns a tensor.

### Aliases:

* tf.compat.v1.keras.backend.dot
* tf.compat.v2.keras.backend.dot
* tf.keras.backend.dot

tf.keras.backend.dot(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

When attempting to multiply a nD tensor with a nD tensor, it reproduces the Theano behavior. (e.g. (2, 3) \* (4, 3, 5) -> (2, 4, 5))

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A tensor, dot product of x and y.

#### Examples:

    # dot product between tensors  
    >>> x = K.placeholder(shape=(2, 3))  
    >>> y = K.placeholder(shape=(3, 4))  
    >>> xy = K.dot(x, y)  
    >>> xy  
    <tf.Tensor 'MatMul\_9:0' shape=(2, 4) dtype=float32>

    # dot product between tensors  
    >>> x = K.placeholder(shape=(32, 28, 3))  
    >>> y = K.placeholder(shape=(3, 4))  
    >>> xy = K.dot(x, y)  
    >>> xy  
    <tf.Tensor 'MatMul\_9:0' shape=(32, 28, 4) dtype=float32>

    # Theano-like behavior example  
    >>> x = K.random\_uniform\_variable(shape=(2, 3), low=0, high=1)  
    >>> y = K.ones((4, 3, 5))  
    >>> xy = K.dot(x, y)  
    >>> K.int\_shape(xy)  
    (2, 4, 5)

# tf.keras.backend.dropout

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dropout#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dropout#aliases)

Sets entries in x to zero at random, while scaling the entire tensor.

### Aliases:

* tf.compat.v1.keras.backend.dropout
* tf.compat.v2.keras.backend.dropout
* tf.keras.backend.dropout

tf.keras.backend.dropout(  
    x,  
    level,  
    noise\_shape=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: tensor
* **level**: fraction of the entries in the tensor that will be set to 0.
* **noise\_shape**: shape for randomly generated keep/drop flags, must be broadcastable to the shape of x
* **seed**: random seed to ensure determinism.

#### Returns:

A tensor.

# tf.keras.backend.dtype

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dtype#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/dtype#aliases)

Returns the dtype of a Keras tensor or variable, as a string.

### Aliases:

* tf.compat.v1.keras.backend.dtype
* tf.compat.v2.keras.backend.dtype
* tf.keras.backend.dtype

tf.keras.backend.dtype(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

String, dtype of x.

#### Examples:

    >>> from keras import backend as K  
    >>> K.dtype(K.placeholder(shape=(2,4,5)))  
    'float32'  
    >>> K.dtype(K.placeholder(shape=(2,4,5), dtype='float32'))  
    'float32'  
    >>> K.dtype(K.placeholder(shape=(2,4,5), dtype='float64'))  
    'float64'  
    # Keras variable  
    >>> kvar = K.variable(np.array([[1, 2], [3, 4]]))  
    >>> K.dtype(kvar)  
    'float32'  
    >>> kvar = K.variable(np.array([[1, 2], [3, 4]]), dtype='float32')  
    >>> K.dtype(kvar)  
    'float32'

# tf.keras.backend.elu

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/elu#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/elu#aliases)

Exponential linear unit.

### Aliases:

* tf.compat.v1.keras.backend.elu
* tf.compat.v2.keras.backend.elu
* tf.keras.backend.elu

tf.keras.backend.elu(  
    x,  
    alpha=1.0  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable to compute the activation function for.
* **alpha**: A scalar, slope of negative section.

#### Returns:

A tensor.

# tf.keras.backend.epsilon

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/epsilon#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/epsilon#aliases)

Returns the value of the fuzz factor used in numeric expressions.

### Aliases:

* tf.compat.v1.keras.backend.epsilon
* tf.compat.v2.keras.backend.epsilon
* tf.keras.backend.epsilon

tf.keras.backend.epsilon()

Defined in [python/keras/backend\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend_config.py).

#### Returns:

A float.

#### Example:

keras.backend.epsilon() >>>1e-07

# tf.keras.backend.equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/equal#aliases)

Element-wise equality between two tensors.

### Aliases:

* tf.compat.v1.keras.backend.equal
* tf.compat.v2.keras.backend.equal
* tf.keras.backend.equal

tf.keras.backend.equal(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A bool tensor.

# tf.keras.backend.eval

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/eval#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/eval#aliases)

Evaluates the value of a variable.

### Aliases:

* tf.compat.v1.keras.backend.eval
* tf.compat.v2.keras.backend.eval
* tf.keras.backend.eval

tf.keras.backend.eval(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A variable.

#### Returns:

A Numpy array.

#### Examples:

    >>> from keras import backend as K  
    >>> kvar = K.variable(np.array([[1, 2], [3, 4]]), dtype='float32')  
    >>> K.eval(kvar)  
    array([[ 1.,  2.],  
           [ 3.,  4.]], dtype=float32)

# tf.keras.backend.exp

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/exp#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/exp#aliases)

Element-wise exponential.

### Aliases:

* tf.compat.v1.keras.backend.exp
* tf.compat.v2.keras.backend.exp
* tf.keras.backend.exp

tf.keras.backend.exp(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.expand\_dims

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/expand_dims#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/expand_dims#aliases)

Adds a 1-sized dimension at index "axis".

### Aliases:

* tf.compat.v1.keras.backend.expand\_dims
* tf.compat.v2.keras.backend.expand\_dims
* tf.keras.backend.expand\_dims

tf.keras.backend.expand\_dims(  
    x,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: Position where to add a new axis.

#### Returns:

A tensor with expanded dimensions.

# tf.keras.backend.eye

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/eye#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/eye#aliases)

Instantiate an identity matrix and returns it.

### Aliases:

* tf.compat.v1.keras.backend.eye
* tf.compat.v2.keras.backend.eye
* tf.keras.backend.eye

tf.keras.backend.eye(  
    size,  
    dtype=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **size**: Integer, number of rows/columns.
* **dtype**: String, data type of returned Keras variable.
* **name**: String, name of returned Keras variable.

#### Returns:

A Keras variable, an identity matrix.

#### Example:

    >>> from keras import backend as K  
    >>> kvar = K.eye(3)  
    >>> K.eval(kvar)  
    array([[ 1.,  0.,  0.],  
           [ 0.,  1.,  0.],  
           [ 0.,  0.,  1.]], dtype=float32)

# tf.keras.backend.flatten

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/flatten#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/flatten#aliases)

Flatten a tensor.

### Aliases:

* tf.compat.v1.keras.backend.flatten
* tf.compat.v2.keras.backend.flatten
* tf.keras.backend.flatten

tf.keras.backend.flatten(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor, reshaped into 1-D

# tf.keras.backend.floatx

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/floatx#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/floatx#aliases)

Returns the default float type, as a string.

### Aliases:

* tf.compat.v1.keras.backend.floatx
* tf.compat.v2.keras.backend.floatx
* tf.keras.backend.floatx

tf.keras.backend.floatx()

Defined in [python/keras/backend\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend_config.py).

E.g. 'float16', 'float32', 'float64'.

#### Returns:

String, the current default float type.

#### Example:

keras.backend.floatx() >>> 'float32'

# tf.keras.backend.foldl

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/foldl#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/foldl#aliases)

Reduce elems using fn to combine them from left to right.

### Aliases:

* tf.compat.v1.keras.backend.foldl
* tf.compat.v2.keras.backend.foldl
* tf.keras.backend.foldl

tf.keras.backend.foldl(  
    fn,  
    elems,  
    initializer=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **fn**: Callable that will be called upon each element in elems and an accumulator, for instance lambda acc, x: acc + x
* **elems**: tensor
* **initializer**: The first value used (elems[0] in case of None)
* **name**: A string name for the foldl node in the graph

#### Returns:

Tensor with same type and shape as initializer.

# tf.keras.backend.foldr

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/foldr#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/foldr#aliases)

Reduce elems using fn to combine them from right to left.

### Aliases:

* tf.compat.v1.keras.backend.foldr
* tf.compat.v2.keras.backend.foldr
* tf.keras.backend.foldr

tf.keras.backend.foldr(  
    fn,  
    elems,  
    initializer=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **fn**: Callable that will be called upon each element in elems and an accumulator, for instance lambda acc, x: acc + x
* **elems**: tensor
* **initializer**: The first value used (elems[-1] in case of None)
* **name**: A string name for the foldr node in the graph

#### Returns:

Same type and shape as initializer

# tf.keras.backend.function

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/function#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/function#aliases)

Instantiates a Keras function.

### Aliases:

* tf.compat.v1.keras.backend.function
* tf.compat.v2.keras.backend.function
* tf.keras.backend.function

tf.keras.backend.function(  
    inputs,  
    outputs,  
    updates=None,  
    name=None,  
    \*\*kwargs  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **inputs**: List of placeholder tensors.
* **outputs**: List of output tensors.
* **updates**: List of update ops.
* **name**: String, name of function.
* **\*\*kwargs**: Passed to tf.Session.run.

#### Returns:

Output values as Numpy arrays.

#### Raises:

* **ValueError**: if invalid kwargs are passed in or if in eager execution.

# tf.keras.backend.gather

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/gather#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/gather#aliases)

Retrieves the elements of indices indices in the tensor reference.

### Aliases:

* tf.compat.v1.keras.backend.gather
* tf.compat.v2.keras.backend.gather
* tf.keras.backend.gather

tf.keras.backend.gather(  
    reference,  
    indices  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **reference**: A tensor.
* **indices**: An integer tensor of indices.

#### Returns:

A tensor of same type as reference.

# tf.keras.backend.get\_uid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_uid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_uid#aliases)

Associates a string prefix with an integer counter in a TensorFlow graph.

### Aliases:

* tf.compat.v1.keras.backend.get\_uid
* tf.compat.v2.keras.backend.get\_uid
* tf.keras.backend.get\_uid

tf.keras.backend.get\_uid(prefix='')

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **prefix**: String prefix to index.

#### Returns:

Unique integer ID.

#### Example:

  >>> get\_uid('dense')  
  1  
  >>> get\_uid('dense')  
  2

# tf.keras.backend.get\_value

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_value#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_value#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/get_value#used_in_the_guide)

Returns the value of a variable.

### Aliases:

* tf.compat.v1.keras.backend.get\_value
* tf.compat.v2.keras.backend.get\_value
* tf.keras.backend.get\_value

tf.keras.backend.get\_value(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

### Used in the guide:

* [Keras custom callbacks](https://www.tensorflow.org/beta/guide/keras/custom_callback)

#### Arguments:

* **x**: input variable.

#### Returns:

A Numpy array.

# tf.keras.backend.gradients

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/gradients#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/gradients#aliases)

Returns the gradients of loss w.r.t. variables.

### Aliases:

* tf.compat.v1.keras.backend.gradients
* tf.compat.v2.keras.backend.gradients
* tf.keras.backend.gradients

tf.keras.backend.gradients(  
    loss,  
    variables  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **loss**: Scalar tensor to minimize.
* **variables**: List of variables.

#### Returns:

A gradients tensor.

# tf.keras.backend.greater

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/greater#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/greater#aliases)

Element-wise truth value of (x > y).

### Aliases:

* tf.compat.v1.keras.backend.greater
* tf.compat.v2.keras.backend.greater
* tf.keras.backend.greater

tf.keras.backend.greater(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A bool tensor.

# tf.keras.backend.greater\_equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/greater_equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/greater_equal#aliases)

Element-wise truth value of (x >= y).

### Aliases:

* tf.compat.v1.keras.backend.greater\_equal
* tf.compat.v2.keras.backend.greater\_equal
* tf.keras.backend.greater\_equal

tf.keras.backend.greater\_equal(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A bool tensor.

# tf.keras.backend.hard\_sigmoid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/hard_sigmoid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/hard_sigmoid#aliases)

Segment-wise linear approximation of sigmoid.

### Aliases:

* tf.compat.v1.keras.backend.hard\_sigmoid
* tf.compat.v2.keras.backend.hard\_sigmoid
* tf.keras.backend.hard\_sigmoid

tf.keras.backend.hard\_sigmoid(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Faster than sigmoid. Returns 0. if x < -2.5, 1. if x > 2.5. In -2.5 <= x <= 2.5, returns 0.2 \* x + 0.5.

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.image\_data\_format

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/image_data_format#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/image_data_format#aliases)

Returns the default image data format convention.

### Aliases:

* tf.compat.v1.keras.backend.image\_data\_format
* tf.compat.v2.keras.backend.image\_data\_format
* tf.keras.backend.image\_data\_format

tf.keras.backend.image\_data\_format()

Defined in [python/keras/backend\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend_config.py).

#### Returns:

A string, either 'channels\_first' or 'channels\_last'

#### Example:

keras.backend.image\_data\_format() >>> 'channels\_first'

# tf.keras.backend.int\_shape

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/int_shape#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/int_shape#aliases)

Returns the shape of tensor or variable as a tuple of int or None entries.

### Aliases:

* tf.compat.v1.keras.backend.int\_shape
* tf.compat.v2.keras.backend.int\_shape
* tf.keras.backend.int\_shape

tf.keras.backend.int\_shape(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tuple of integers (or None entries).

#### Examples:

    >>> from keras import backend as K  
    >>> input = K.placeholder(shape=(2, 4, 5))  
    >>> K.int\_shape(input)  
    (2, 4, 5)  
    >>> val = np.array([[1, 2], [3, 4]])  
    >>> kvar = K.variable(value=val)  
    >>> K.int\_shape(kvar)  
    (2, 2)

# tf.keras.backend.in\_test\_phase

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_test_phase#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_test_phase#aliases)

Selects x in test phase, and alt otherwise.

### Aliases:

* tf.compat.v1.keras.backend.in\_test\_phase
* tf.compat.v2.keras.backend.in\_test\_phase
* tf.keras.backend.in\_test\_phase

tf.keras.backend.in\_test\_phase(  
    x,  
    alt,  
    training=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Note that alt should have the same shape as x.

#### Arguments:

* **x**: What to return in test phase (tensor or callable that returns a tensor).
* **alt**: What to return otherwise (tensor or callable that returns a tensor).
* **training**: Optional scalar tensor (or Python boolean, or Python integer) specifying the learning phase.

#### Returns:

Either x or alt based on K.learning\_phase.

# tf.keras.backend.in\_top\_k

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_top_k#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_top_k#aliases)

Returns whether the targets are in the top k predictions.

### Aliases:

* tf.compat.v1.keras.backend.in\_top\_k
* tf.compat.v2.keras.backend.in\_top\_k
* tf.keras.backend.in\_top\_k

tf.keras.backend.in\_top\_k(  
    predictions,  
    targets,  
    k  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **predictions**: A tensor of shape (batch\_size, classes) and type float32.
* **targets**: A 1D tensor of length batch\_size and type int32 or int64.
* **k**: An int, number of top elements to consider.

#### Returns:

A 1D tensor of length batch\_size and type bool. output[i] is True if predictions[i, targets[i]] is within top-k values of predictions[i].

# tf.keras.backend.in\_train\_phase

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_train_phase#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/in_train_phase#aliases)

Selects x in train phase, and alt otherwise.

### Aliases:

* tf.compat.v1.keras.backend.in\_train\_phase
* tf.compat.v2.keras.backend.in\_train\_phase
* tf.keras.backend.in\_train\_phase

tf.keras.backend.in\_train\_phase(  
    x,  
    alt,  
    training=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Note that alt should have the same shape as x.

#### Arguments:

* **x**: What to return in train phase (tensor or callable that returns a tensor).
* **alt**: What to return otherwise (tensor or callable that returns a tensor).
* **training**: Optional scalar tensor (or Python boolean, or Python integer) specifying the learning phase.

#### Returns:

Either x or alt based on the training flag. the training flag defaults to K.learning\_phase().

# tf.keras.backend.is\_sparse

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/is_sparse#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/is_sparse#aliases)

Returns whether a tensor is a sparse tensor.

### Aliases:

* tf.compat.v1.keras.backend.is\_sparse
* tf.compat.v2.keras.backend.is\_sparse
* tf.keras.backend.is\_sparse

tf.keras.backend.is\_sparse(tensor)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **tensor**: A tensor instance.

#### Returns:

A boolean.

#### Example:

    >>> from keras import backend as K  
    >>> a = K.placeholder((2, 2), sparse=False)  
    >>> print(K.is\_sparse(a))  
    False  
    >>> b = K.placeholder((2, 2), sparse=True)  
    >>> print(K.is\_sparse(b))  
    True

# tf.keras.backend.l2\_normalize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/l2_normalize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/l2_normalize#aliases)

Normalizes a tensor wrt the L2 norm alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.l2\_normalize
* tf.compat.v2.keras.backend.l2\_normalize
* tf.keras.backend.l2\_normalize

tf.keras.backend.l2\_normalize(  
    x,  
    axis=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **axis**: axis along which to perform normalization.

#### Returns:

A tensor.

# tf.keras.backend.learning\_phase

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/learning_phase#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/learning_phase#aliases)

Returns the learning phase flag.

### Aliases:

* tf.compat.v1.keras.backend.learning\_phase
* tf.compat.v2.keras.backend.learning\_phase
* tf.keras.backend.learning\_phase

tf.keras.backend.learning\_phase()

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

The learning phase flag is a bool tensor (0 = test, 1 = train) to be passed as input to any Keras function that uses a different behavior at train time and test time.

#### Returns:

Learning phase (scalar integer tensor or Python integer).

# tf.keras.backend.learning\_phase\_scope

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/learning_phase_scope#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/learning_phase_scope#aliases)

Provides a scope within which the learning phase is equal to value.

### Aliases:

* tf.compat.v1.keras.backend.learning\_phase\_scope
* tf.compat.v2.keras.backend.learning\_phase\_scope
* tf.keras.backend.learning\_phase\_scope

tf.keras.backend.learning\_phase\_scope(value)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

The learning phase gets restored to its original value upon exiting the scope.

#### Arguments:

* **value**: Learning phase value, either 0 or 1 (integers).

#### Yields:

None.

#### Raises:

* **ValueError**: if value is neither 0 nor 1.

# tf.keras.backend.less

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/less#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/less#aliases)

Element-wise truth value of (x < y).

### Aliases:

* tf.compat.v1.keras.backend.less
* tf.compat.v2.keras.backend.less
* tf.keras.backend.less

tf.keras.backend.less(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A bool tensor.

# tf.keras.backend.less\_equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/less_equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/less_equal#aliases)

Element-wise truth value of (x <= y).

### Aliases:

* tf.compat.v1.keras.backend.less\_equal
* tf.compat.v2.keras.backend.less\_equal
* tf.keras.backend.less\_equal

tf.keras.backend.less\_equal(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A bool tensor.

# tf.keras.backend.local\_conv1d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/local_conv1d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/local_conv1d#aliases)

Apply 1D conv with un-shared weights.

### Aliases:

* tf.compat.v1.keras.backend.local\_conv1d
* tf.compat.v2.keras.backend.local\_conv1d
* tf.keras.backend.local\_conv1d

tf.keras.backend.local\_conv1d(  
    inputs,  
    kernel,  
    kernel\_size,  
    strides,  
    data\_format=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **inputs**: 3D tensor with shape: (batch\_size, steps, input\_dim) if data\_format is "channels\_last" or (batch\_size, input\_dim, steps) if data\_format is "channels\_first".
* **kernel**: the unshared weight for convolution, with shape (output\_length, feature\_dim, filters).
* **kernel\_size**: a tuple of a single integer, specifying the length of the 1D convolution window.
* **strides**: a tuple of a single integer, specifying the stride length of the convolution.
* **data\_format**: the data format, channels\_first or channels\_last.

#### Returns:

A 3d tensor with shape: (batch\_size, output\_length, filters) if data\_format='channels\_first' or 3D tensor with shape: (batch\_size, filters, output\_length) if data\_format='channels\_last'.

# tf.keras.backend.local\_conv2d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/local_conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/local_conv2d#aliases)

Apply 2D conv with un-shared weights.

### Aliases:

* tf.compat.v1.keras.backend.local\_conv2d
* tf.compat.v2.keras.backend.local\_conv2d
* tf.keras.backend.local\_conv2d

tf.keras.backend.local\_conv2d(  
    inputs,  
    kernel,  
    kernel\_size,  
    strides,  
    output\_shape,  
    data\_format=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **inputs**: 4D tensor with shape: (batch\_size, filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4D tensor with shape: (batch\_size, new\_rows, new\_cols, filters) if data\_format='channels\_last'.
* **kernel**: the unshared weight for convolution, with shape (output\_items, feature\_dim, filters).
* **kernel\_size**: a tuple of 2 integers, specifying the width and height of the 2D convolution window.
* **strides**: a tuple of 2 integers, specifying the strides of the convolution along the width and height.
* **output\_shape**: a tuple with (output\_row, output\_col).
* **data\_format**: the data format, channels\_first or channels\_last.

#### Returns:

A 4D tensor with shape: (batch\_size, filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4D tensor with shape: (batch\_size, new\_rows, new\_cols, filters) if data\_format='channels\_last'.

# tf.keras.backend.log

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/log#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/log#aliases)

Element-wise log.

### Aliases:

* tf.compat.v1.keras.backend.log
* tf.compat.v2.keras.backend.log
* tf.keras.backend.log

tf.keras.backend.log(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.manual\_variable\_initialization

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/manual_variable_initialization#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/manual_variable_initialization#aliases)

Sets the manual variable initialization flag.

### Aliases:

* tf.compat.v1.keras.backend.manual\_variable\_initialization
* tf.compat.v2.keras.backend.manual\_variable\_initialization
* tf.keras.backend.manual\_variable\_initialization

tf.keras.backend.manual\_variable\_initialization(value)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

This boolean flag determines whether variables should be initialized as they are instantiated (default), or if the user should handle the initialization (e.g. via tf.compat.v1.initialize\_all\_variables()).

#### Arguments:

* **value**: Python boolean.

# tf.keras.backend.map\_fn

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/map_fn#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/map_fn#aliases)

Map the function fn over the elements elems and return the outputs.

### Aliases:

* tf.compat.v1.keras.backend.map\_fn
* tf.compat.v2.keras.backend.map\_fn
* tf.keras.backend.map\_fn

tf.keras.backend.map\_fn(  
    fn,  
    elems,  
    name=None,  
    dtype=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **fn**: Callable that will be called upon each element in elems
* **elems**: tensor
* **name**: A string name for the map node in the graph
* **dtype**: Output data type.

#### Returns:

Tensor with dtype dtype.

# tf.keras.backend.max

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/max#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/max#aliases)

Maximum value in a tensor.

### Aliases:

* tf.compat.v1.keras.backend.max
* tf.compat.v2.keras.backend.max
* tf.keras.backend.max

tf.keras.backend.max(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to find maximum values.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1. If keepdims is True, the reduced dimension is retained with length 1.

#### Returns:

A tensor with maximum values of x.

# tf.keras.backend.maximum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/maximum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/maximum#aliases)

Element-wise maximum of two tensors.

### Aliases:

* tf.compat.v1.keras.backend.maximum
* tf.compat.v2.keras.backend.maximum
* tf.keras.backend.maximum

tf.keras.backend.maximum(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.mean

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/mean#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/mean#aliases)

Mean of a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.mean
* tf.compat.v2.keras.backend.mean
* tf.keras.backend.mean

tf.keras.backend.mean(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: A list of integer. Axes to compute the mean.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is True, the reduced dimensions are retained with length 1.

#### Returns:

A tensor with the mean of elements of x.

# tf.keras.backend.min

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/min#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/min#aliases)

Minimum value in a tensor.

### Aliases:

* tf.compat.v1.keras.backend.min
* tf.compat.v2.keras.backend.min
* tf.keras.backend.min

tf.keras.backend.min(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to find minimum values.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1. If keepdims is True, the reduced dimension is retained with length 1.

#### Returns:

A tensor with minimum values of x.

# tf.keras.backend.minimum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/minimum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/minimum#aliases)

Element-wise minimum of two tensors.

### Aliases:

* tf.compat.v1.keras.backend.minimum
* tf.compat.v2.keras.backend.minimum
* tf.keras.backend.minimum

tf.keras.backend.minimum(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.moving\_average\_update

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/moving_average_update#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/moving_average_update#aliases)

Compute the moving average of a variable.

### Aliases:

* tf.compat.v1.keras.backend.moving\_average\_update
* tf.compat.v2.keras.backend.moving\_average\_update
* tf.keras.backend.moving\_average\_update

tf.keras.backend.moving\_average\_update(  
    x,  
    value,  
    momentum  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A Variable.
* **value**: A tensor with the same shape as variable.
* **momentum**: The moving average momentum.

#### Returns:

An Operation to update the variable.

# tf.keras.backend.name\_scope

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/name_scope#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/name_scope#aliases)

A context manager for use when defining a Python op.

### Aliases:

* tf.compat.v2.keras.backend.name\_scope
* tf.keras.backend.name\_scope

tf.keras.backend.name\_scope(name)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

This context manager pushes a name scope, which will make the name of all operations added within it have a prefix.

For example, to define a new Python op called my\_op:

def my\_op(a):  
  with tf.name\_scope("MyOp") as scope:  
    a = tf.convert\_to\_tensor(a, name="a")  
    # Define some computation that uses `a`.  
    return foo\_op(..., name=scope)

When executed, the Tensor a will have the name MyOp/a.

#### Args:

* **name**: The prefix to use on all names created within the name scope.

#### Returns:

Name scope context manager.

# tf.keras.backend.ndim

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ndim#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ndim#aliases)

Returns the number of axes in a tensor, as an integer.

### Aliases:

* tf.compat.v1.keras.backend.ndim
* tf.compat.v2.keras.backend.ndim
* tf.keras.backend.ndim

tf.keras.backend.ndim(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

Integer (scalar), number of axes.

#### Examples:

    >>> from keras import backend as K  
    >>> input = K.placeholder(shape=(2, 4, 5))  
    >>> val = np.array([[1, 2], [3, 4]])  
    >>> kvar = K.variable(value=val)  
    >>> K.ndim(input)  
    3  
    >>> K.ndim(kvar)  
    2

# tf.keras.backend.normalize\_batch\_in\_training

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/normalize_batch_in_training#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/normalize_batch_in_training#aliases)

Computes mean and std for batch then apply batch\_normalization on batch.

### Aliases:

* tf.compat.v1.keras.backend.normalize\_batch\_in\_training
* tf.compat.v2.keras.backend.normalize\_batch\_in\_training
* tf.keras.backend.normalize\_batch\_in\_training

tf.keras.backend.normalize\_batch\_in\_training(  
    x,  
    gamma,  
    beta,  
    reduction\_axes,  
    epsilon=0.001  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Input tensor or variable.
* **gamma**: Tensor by which to scale the input.
* **beta**: Tensor with which to center the input.
* **reduction\_axes**: iterable of integers, axes over which to normalize.
* **epsilon**: Fuzz factor.

#### Returns:

A tuple length of 3, (normalized\_tensor, mean, variance).

# tf.keras.backend.not\_equal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/not_equal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/not_equal#aliases)

Element-wise inequality between two tensors.

### Aliases:

* tf.compat.v1.keras.backend.not\_equal
* tf.compat.v2.keras.backend.not\_equal
* tf.keras.backend.not\_equal

tf.keras.backend.not\_equal(  
    x,  
    y  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **y**: Tensor or variable.

#### Returns:

A bool tensor.

# tf.keras.backend.ones

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ones#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ones#aliases)

Instantiates an all-ones variable and returns it.

### Aliases:

* tf.compat.v1.keras.backend.ones
* tf.compat.v2.keras.backend.ones
* tf.keras.backend.ones

tf.keras.backend.ones(  
    shape,  
    dtype=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **shape**: Tuple of integers, shape of returned Keras variable.
* **dtype**: String, data type of returned Keras variable.
* **name**: String, name of returned Keras variable.

#### Returns:

A Keras variable, filled with 1.0. Note that if shape was symbolic, we cannot return a variable, and will return a dynamically-shaped tensor instead.

#### Example:

    >>> from keras import backend as K  
    >>> kvar = K.ones((3,4))  
    >>> K.eval(kvar)  
    array([[ 1.,  1.,  1.,  1.],  
           [ 1.,  1.,  1.,  1.],  
           [ 1.,  1.,  1.,  1.]], dtype=float32)

# tf.keras.backend.ones\_like

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ones_like#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/ones_like#aliases)

Instantiates an all-ones variable of the same shape as another tensor.

### Aliases:

* tf.compat.v1.keras.backend.ones\_like
* tf.compat.v2.keras.backend.ones\_like
* tf.keras.backend.ones\_like

tf.keras.backend.ones\_like(  
    x,  
    dtype=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Keras variable or tensor.
* **dtype**: String, dtype of returned Keras variable. None uses the dtype of x.
* **name**: String, name for the variable to create.

#### Returns:

A Keras variable with the shape of x filled with ones.

#### Example:

    >>> from keras import backend as K  
    >>> kvar = K.variable(np.random.random((2,3)))  
    >>> kvar\_ones = K.ones\_like(kvar)  
    >>> K.eval(kvar\_ones)  
    array([[ 1.,  1.,  1.],  
           [ 1.,  1.,  1.]], dtype=float32)

# tf.keras.backend.one\_hot

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/one_hot#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/one_hot#aliases)

Computes the one-hot representation of an integer tensor.

### Aliases:

* tf.compat.v1.keras.backend.one\_hot
* tf.compat.v2.keras.backend.one\_hot
* tf.keras.backend.one\_hot

tf.keras.backend.one\_hot(  
    indices,  
    num\_classes  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **indices**: nD integer tensor of shape (batch\_size, dim1, dim2, ... dim(n-1))
* **num\_classes**: Integer, number of classes to consider.

#### Returns:

(n + 1)D one hot representation of the input with shape (batch\_size, dim1, dim2, ... dim(n-1), num\_classes)

#### Returns:

The one-hot tensor.

# tf.keras.backend.permute\_dimensions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/permute_dimensions#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/permute_dimensions#aliases)

Permutes axes in a tensor.

### Aliases:

* tf.compat.v1.keras.backend.permute\_dimensions
* tf.compat.v2.keras.backend.permute\_dimensions
* tf.keras.backend.permute\_dimensions

tf.keras.backend.permute\_dimensions(  
    x,  
    pattern  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **pattern**: A tuple of dimension indices, e.g. (0, 2, 1).

#### Returns:

A tensor.

# tf.keras.backend.placeholder

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/placeholder#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/placeholder#aliases)

Instantiates a placeholder tensor and returns it.

### Aliases:

* tf.compat.v1.keras.backend.placeholder
* tf.compat.v2.keras.backend.placeholder
* tf.keras.backend.placeholder

tf.keras.backend.placeholder(  
    shape=None,  
    ndim=None,  
    dtype=None,  
    sparse=False,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **shape**: Shape of the placeholder (integer tuple, may include None entries).
* **ndim**: Number of axes of the tensor. At least one of {shape, ndim} must be specified. If both are specified, shape is used.
* **dtype**: Placeholder type.
* **sparse**: Boolean, whether the placeholder should have a sparse type.
* **name**: Optional name string for the placeholder.

#### Raises:

* **ValueError**: If called with eager execution.

#### Returns:

Tensor instance (with Keras metadata included).

#### Examples:

    >>> from keras import backend as K  
    >>> input\_ph = K.placeholder(shape=(2, 4, 5))  
    >>> input\_ph  
    <tf.Tensor 'Placeholder\_4:0' shape=(2, 4, 5) dtype=float32>

# tf.keras.backend.pool2d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pool2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pool2d#aliases)

2D Pooling.

### Aliases:

* tf.compat.v1.keras.backend.pool2d
* tf.compat.v2.keras.backend.pool2d
* tf.keras.backend.pool2d

tf.keras.backend.pool2d(  
    x,  
    pool\_size,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    pool\_mode='max'  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **pool\_size**: tuple of 2 integers.
* **strides**: tuple of 2 integers.
* **padding**: string, "same" or "valid".
* **data\_format**: string, "channels\_last" or "channels\_first".
* **pool\_mode**: string, "max" or "avg".

#### Returns:

A tensor, result of 2D pooling.

#### Raises:

* **ValueError**: if data\_format is neither "channels\_last" or "channels\_first".
* **ValueError**: if pool\_size is not a tuple of 2 integers.
* **ValueError**: if strides is not a tuple of 2 integers.
* **ValueError**: if pool\_mode is neither "max" or "avg".

# tf.keras.backend.pool3d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pool3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pool3d#aliases)

3D Pooling.

### Aliases:

* tf.compat.v1.keras.backend.pool3d
* tf.compat.v2.keras.backend.pool3d
* tf.keras.backend.pool3d

tf.keras.backend.pool3d(  
    x,  
    pool\_size,  
    strides=(1, 1, 1),  
    padding='valid',  
    data\_format=None,  
    pool\_mode='max'  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **pool\_size**: tuple of 3 integers.
* **strides**: tuple of 3 integers.
* **padding**: string, "same" or "valid".
* **data\_format**: string, "channels\_last" or "channels\_first".
* **pool\_mode**: string, "max" or "avg".

#### Returns:

A tensor, result of 3D pooling.

#### Raises:

* **ValueError**: if data\_format is neither "channels\_last" or "channels\_first".
* **ValueError**: if pool\_mode is neither "max" or "avg".

# tf.keras.backend.pow

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pow#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/pow#aliases)

Element-wise exponentiation.

### Aliases:

* tf.compat.v1.keras.backend.pow
* tf.compat.v2.keras.backend.pow
* tf.keras.backend.pow

tf.keras.backend.pow(  
    x,  
    a  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **a**: Python integer.

#### Returns:

A tensor.

# tf.keras.backend.print\_tensor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/print_tensor#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/print_tensor#aliases)

Prints message and the tensor value when evaluated.

### Aliases:

* tf.compat.v1.keras.backend.print\_tensor
* tf.compat.v2.keras.backend.print\_tensor
* tf.keras.backend.print\_tensor

tf.keras.backend.print\_tensor(  
    x,  
    message=''  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Note that print\_tensor returns a new tensor identical to x which should be used in the following code. Otherwise the print operation is not taken into account during evaluation.

#### Example:

   >>> x = K.print\_tensor(x, message="x is: ")

#### Arguments:

* **x**: Tensor to print.
* **message**: Message to print jointly with the tensor.

#### Returns:

The same tensor x, unchanged.

# tf.keras.backend.prod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/prod#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/prod#aliases)

Multiplies the values in a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.prod
* tf.compat.v2.keras.backend.prod
* tf.keras.backend.prod

tf.keras.backend.prod(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to compute the product.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1. If keepdims is True, the reduced dimension is retained with length 1.

#### Returns:

A tensor with the product of elements of x.

# tf.keras.backend.random\_binomial

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_binomial#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_binomial#aliases)

Returns a tensor with random binomial distribution of values.

### Aliases:

* tf.compat.v1.keras.backend.random\_binomial
* tf.compat.v2.keras.backend.random\_binomial
* tf.keras.backend.random\_binomial

tf.keras.backend.random\_binomial(  
    shape,  
    p=0.0,  
    dtype=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

The binomial distribution with parameters n and p is the probability distribution of the number of successful Bernoulli process. Only supports n = 1 for now.

#### Arguments:

* **shape**: A tuple of integers, the shape of tensor to create.
* **p**: A float, 0. <= p <= 1, probability of binomial distribution.
* **dtype**: String, dtype of returned tensor.
* **seed**: Integer, random seed.

#### Returns:

A tensor.

# tf.keras.backend.random\_normal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal#used_in_the_guide)

Returns a tensor with normal distribution of values.

### Aliases:

* tf.compat.v1.keras.backend.random\_normal
* tf.compat.v2.keras.backend.random\_normal
* tf.keras.backend.random\_normal

tf.keras.backend.random\_normal(  
    shape,  
    mean=0.0,  
    stddev=1.0,  
    dtype=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

### Used in the guide:

* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

#### Arguments:

* **shape**: A tuple of integers, the shape of tensor to create.
* **mean**: A float, mean of the normal distribution to draw samples.
* **stddev**: A float, standard deviation of the normal distribution to draw samples.
* **dtype**: String, dtype of returned tensor.
* **seed**: Integer, random seed.

#### Returns:

A tensor.

# tf.keras.backend.random\_normal\_variable

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal_variable#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_normal_variable#aliases)

Instantiates a variable with values drawn from a normal distribution.

### Aliases:

* tf.compat.v1.keras.backend.random\_normal\_variable
* tf.compat.v2.keras.backend.random\_normal\_variable
* tf.keras.backend.random\_normal\_variable

tf.keras.backend.random\_normal\_variable(  
    shape,  
    mean,  
    scale,  
    dtype=None,  
    name=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **shape**: Tuple of integers, shape of returned Keras variable.
* **mean**: Float, mean of the normal distribution.
* **scale**: Float, standard deviation of the normal distribution.
* **dtype**: String, dtype of returned Keras variable.
* **name**: String, name of returned Keras variable.
* **seed**: Integer, random seed.

#### Returns:

A Keras variable, filled with drawn samples.

#### Example:

    # TensorFlow example  
    >>> kvar = K.random\_normal\_variable((2,3), 0, 1)  
    >>> kvar  
    <tensorflow.python.ops.variables.Variable object at 0x10ab12dd0>  
    >>> K.eval(kvar)  
    array([[ 1.19591331,  0.68685907, -0.63814116],  
           [ 0.92629528,  0.28055015,  1.70484698]], dtype=float32)

# tf.keras.backend.random\_uniform

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_uniform#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_uniform#aliases)

Returns a tensor with uniform distribution of values.

### Aliases:

* tf.compat.v1.keras.backend.random\_uniform
* tf.compat.v2.keras.backend.random\_uniform
* tf.keras.backend.random\_uniform

tf.keras.backend.random\_uniform(  
    shape,  
    minval=0.0,  
    maxval=1.0,  
    dtype=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **shape**: A tuple of integers, the shape of tensor to create.
* **minval**: A float, lower boundary of the uniform distribution to draw samples.
* **maxval**: A float, upper boundary of the uniform distribution to draw samples.
* **dtype**: String, dtype of returned tensor.
* **seed**: Integer, random seed.

#### Returns:

A tensor.

# tf.keras.backend.random\_uniform\_variable

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_uniform_variable#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/random_uniform_variable#aliases)

Instantiates a variable with values drawn from a uniform distribution.

### Aliases:

* tf.compat.v1.keras.backend.random\_uniform\_variable
* tf.compat.v2.keras.backend.random\_uniform\_variable
* tf.keras.backend.random\_uniform\_variable

tf.keras.backend.random\_uniform\_variable(  
    shape,  
    low,  
    high,  
    dtype=None,  
    name=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **shape**: Tuple of integers, shape of returned Keras variable.
* **low**: Float, lower boundary of the output interval.
* **high**: Float, upper boundary of the output interval.
* **dtype**: String, dtype of returned Keras variable.
* **name**: String, name of returned Keras variable.
* **seed**: Integer, random seed.

#### Returns:

A Keras variable, filled with drawn samples.

#### Example:

    # TensorFlow example  
    >>> kvar = K.random\_uniform\_variable((2,3), 0, 1)  
    >>> kvar  
    <tensorflow.python.ops.variables.Variable object at 0x10ab40b10>  
    >>> K.eval(kvar)  
    array([[ 0.10940075,  0.10047495,  0.476143  ],  
           [ 0.66137183,  0.00869417,  0.89220798]], dtype=float32)

# tf.keras.backend.relu

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/relu#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/relu#aliases)

Rectified linear unit.

### Aliases:

* tf.compat.v1.keras.backend.relu
* tf.compat.v2.keras.backend.relu
* tf.keras.backend.relu

tf.keras.backend.relu(  
    x,  
    alpha=0.0,  
    max\_value=None,  
    threshold=0  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

With default values, it returns element-wise max(x, 0).

Otherwise, it follows: f(x) = max\_value for x >= max\_value, f(x) = x for threshold <= x < max\_value, f(x) = alpha \* (x - threshold) otherwise.

#### Arguments:

* **x**: A tensor or variable.
* **alpha**: A scalar, slope of negative section (default=0.).
* **max\_value**: float. Saturation threshold.
* **threshold**: float. Threshold value for thresholded activation.

#### Returns:

A tensor.

# tf.keras.backend.repeat

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/repeat#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/repeat#aliases)

Repeats a 2D tensor.

### Aliases:

* tf.compat.v1.keras.backend.repeat
* tf.compat.v2.keras.backend.repeat
* tf.keras.backend.repeat

tf.keras.backend.repeat(  
    x,  
    n  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

if x has shape (samples, dim) and n is 2, the output will have shape (samples, 2, dim).

#### Arguments:

* **x**: Tensor or variable.
* **n**: Python integer, number of times to repeat.

#### Returns:

A tensor.

# tf.keras.backend.repeat\_elements

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/repeat_elements#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/repeat_elements#aliases)

Repeats the elements of a tensor along an axis, like np.repeat.

### Aliases:

* tf.compat.v1.keras.backend.repeat\_elements
* tf.compat.v2.keras.backend.repeat\_elements
* tf.keras.backend.repeat\_elements

tf.keras.backend.repeat\_elements(  
    x,  
    rep,  
    axis  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

If x has shape (s1, s2, s3) and axis is 1, the output will have shape (s1, s2 \* rep, s3).

#### Arguments:

* **x**: Tensor or variable.
* **rep**: Python integer, number of times to repeat.
* **axis**: Axis along which to repeat.

#### Returns:

A tensor.

tf.keras.backend.reset\_uids

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reset_uids#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reset_uids#aliases)

Resets graph identifiers.

Aliases:

* tf.compat.v1.keras.backend.reset\_uids
* tf.compat.v2.keras.backend.reset\_uids
* tf.keras.backend.reset\_uids

tf.keras.backend.reset\_uids()

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

# tf.keras.backend.reshape

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reshape#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reshape#aliases)

Reshapes a tensor to the specified shape.

### Aliases:

* tf.compat.v1.keras.backend.reshape
* tf.compat.v2.keras.backend.reshape
* tf.keras.backend.reshape

tf.keras.backend.reshape(  
    x,  
    shape  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **shape**: Target shape tuple.

#### Returns:

A tensor.

# tf.keras.backend.resize\_images

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/resize_images#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/resize_images#aliases)

Resizes the images contained in a 4D tensor.

### Aliases:

* tf.compat.v1.keras.backend.resize\_images
* tf.compat.v2.keras.backend.resize\_images
* tf.keras.backend.resize\_images

tf.keras.backend.resize\_images(  
    x,  
    height\_factor,  
    width\_factor,  
    data\_format,  
    interpolation='nearest'  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable to resize.
* **height\_factor**: Positive integer.
* **width\_factor**: Positive integer.
* **data\_format**: One of "channels\_first", "channels\_last".
* **interpolation**: A string, one of nearest or bilinear.

#### Returns:

A tensor.

#### Raises:

* **ValueError**: in case of incorrect value for data\_format or interpolation.

# tf.keras.backend.resize\_volumes

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/resize_volumes#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/resize_volumes#aliases)

Resizes the volume contained in a 5D tensor.

### Aliases:

* tf.compat.v1.keras.backend.resize\_volumes
* tf.compat.v2.keras.backend.resize\_volumes
* tf.keras.backend.resize\_volumes

tf.keras.backend.resize\_volumes(  
    x,  
    depth\_factor,  
    height\_factor,  
    width\_factor,  
    data\_format  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable to resize.
* **depth\_factor**: Positive integer.
* **height\_factor**: Positive integer.
* **width\_factor**: Positive integer.
* **data\_format**: One of "channels\_first", "channels\_last".

#### Returns:

A tensor.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.reverse

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reverse#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/reverse#aliases)

Reverse a tensor along the specified axes.

### Aliases:

* tf.compat.v1.keras.backend.reverse
* tf.compat.v2.keras.backend.reverse
* tf.keras.backend.reverse

tf.keras.backend.reverse(  
    x,  
    axes  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor to reverse.
* **axes**: Integer or iterable of integers. Axes to reverse.

#### Returns:

A tensor.

# tf.keras.backend.rnn

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/rnn#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/rnn#aliases)

Iterates over the time dimension of a tensor.

### Aliases:

* tf.compat.v1.keras.backend.rnn
* tf.compat.v2.keras.backend.rnn
* tf.keras.backend.rnn

tf.keras.backend.rnn(  
    step\_function,  
    inputs,  
    initial\_states,  
    go\_backwards=False,  
    mask=None,  
    constants=None,  
    unroll=False,  
    input\_length=None,  
    time\_major=False,  
    zero\_output\_for\_mask=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **step\_function**: RNN step function. Args; input; Tensor with shape (samples, ...) (no time dimension), representing input for the batch of samples at a certain time step. states; List of tensors. Returns; output; Tensor with shape (samples, output\_dim) (no time dimension). new\_states; List of tensors, same length and shapes as 'states'. The first state in the list must be the output tensor at the previous timestep.
* **inputs**: Tensor of temporal data of shape (samples, time, ...) (at least 3D), or nested tensors, and each of which has shape (samples, time, ...).
* **initial\_states**: Tensor with shape (samples, state\_size) (no time dimension), containing the initial values for the states used in the step function. In the case that state\_size is in a nested shape, the shape of initial\_states will also follow the nested structure.
* **go\_backwards**: Boolean. If True, do the iteration over the time dimension in reverse order and return the reversed sequence.
* **mask**: Binary tensor with shape (samples, time, 1), with a zero for every element that is masked.
* **constants**: List of constant values passed at each step.
* **unroll**: Whether to unroll the RNN or to use a symbolic while\_loop.
* **input\_length**: If specified, assume time dimension is of this length.
* **time\_major**: Boolean. If true, the inputs and outputs will be in shape (timesteps, batch, ...), whereas in the False case, it will be (batch, timesteps, ...). Using time\_major = True is a bit more efficient because it avoids transposes at the beginning and end of the RNN calculation. However, most TensorFlow data is batch-major, so by default this function accepts input and emits output in batch-major form.
* **zero\_output\_for\_mask**: Boolean. If True, the output for masked timestep will be zeros, whereas in the False case, output from previous timestep is returned.

#### Returns:

A tuple, (last\_output, outputs, new\_states). last\_output: the latest output of the rnn, of shape (samples, ...) outputs: tensor with shape (samples, time, ...) where each entry outputs[s, t] is the output of the step function at time t for sample s. new\_states: list of tensors, latest states returned by the step function, of shape (samples, ...).

#### Raises:

* **ValueError**: if input dimension is less than 3.
* **ValueError**: if unroll is True but input timestep is not a fixed number.
* **ValueError**: if mask is provided (not None) but states is not provided (len(states) == 0).

# tf.keras.backend.round

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/round#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/round#aliases)

Element-wise rounding to the closest integer.

### Aliases:

* tf.compat.v1.keras.backend.round
* tf.compat.v2.keras.backend.round
* tf.keras.backend.round

tf.keras.backend.round(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

In case of tie, the rounding mode used is "half to even".

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.separable\_conv2d

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/separable_conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/separable_conv2d#aliases)

2D convolution with separable filters.

### Aliases:

* tf.compat.v1.keras.backend.separable\_conv2d
* tf.compat.v2.keras.backend.separable\_conv2d
* tf.keras.backend.separable\_conv2d

tf.keras.backend.separable\_conv2d(  
    x,  
    depthwise\_kernel,  
    pointwise\_kernel,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1)  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: input tensor
* **depthwise\_kernel**: convolution kernel for the depthwise convolution.
* **pointwise\_kernel**: kernel for the 1x1 convolution.
* **strides**: strides tuple (length 2).
* **padding**: string, "same" or "valid".
* **data\_format**: string, "channels\_last" or "channels\_first".
* **dilation\_rate**: tuple of integers, dilation rates for the separable convolution.

#### Returns:

Output tensor.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.
* **ValueError**: if strides is not a tuple of 2 integers.

# tf.keras.backend.set\_epsilon

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_epsilon#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_epsilon#aliases)

Sets the value of the fuzz factor used in numeric expressions.

### Aliases:

* tf.compat.v1.keras.backend.set\_epsilon
* tf.compat.v2.keras.backend.set\_epsilon
* tf.keras.backend.set\_epsilon

tf.keras.backend.set\_epsilon(value)

Defined in [python/keras/backend\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend_config.py).

#### Arguments:

* **value**: float. New value of epsilon. Example: python from keras import backend as K K.epsilon() >>> 1e-07 K.set\_epsilon(1e-05) K.epsilon() >>> 1e-05

# tf.keras.backend.set\_floatx

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_floatx#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_floatx#aliases)

Sets the default float type.

### Aliases:

* tf.compat.v1.keras.backend.set\_floatx
* tf.compat.v2.keras.backend.set\_floatx
* tf.keras.backend.set\_floatx

tf.keras.backend.set\_floatx(value)

Defined in [python/keras/backend\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend_config.py).

#### Arguments:

* **value**: String; 'float16', 'float32', or 'float64'. Example: python from keras import backend as K K.floatx() >>> 'float32' K.set\_floatx('float16') K.floatx() >>> 'float16'

#### Raises:

* **ValueError**: In case of invalid value.

# tf.keras.backend.set\_image\_data\_format

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_image_data_format#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_image_data_format#aliases)

Sets the value of the image data format convention.

### Aliases:

* tf.compat.v1.keras.backend.set\_image\_data\_format
* tf.compat.v2.keras.backend.set\_image\_data\_format
* tf.keras.backend.set\_image\_data\_format

tf.keras.backend.set\_image\_data\_format(data\_format)

Defined in [python/keras/backend\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend_config.py).

#### Arguments:

* **data\_format**: string. 'channels\_first' or 'channels\_last'. Example: python from keras import backend as K K.image\_data\_format() >>> 'channels\_first' K.set\_image\_data\_format('channels\_last') K.image\_data\_format() >>> 'channels\_last'

#### Raises:

* **ValueError**: In case of invalid data\_format value.

# tf.keras.backend.set\_learning\_phase

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_learning_phase#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_learning_phase#aliases)

Sets the learning phase to a fixed value.

### Aliases:

* tf.compat.v1.keras.backend.set\_learning\_phase
* tf.compat.v2.keras.backend.set\_learning\_phase
* tf.keras.backend.set\_learning\_phase

tf.keras.backend.set\_learning\_phase(value)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **value**: Learning phase value, either 0 or 1 (integers).

#### Raises:

* **ValueError**: if value is neither 0 nor 1.

# tf.keras.backend.set\_value

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_value#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_value#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/set_value#used_in_the_guide)

Sets the value of a variable, from a Numpy array.

### Aliases:

* tf.compat.v1.keras.backend.set\_value
* tf.compat.v2.keras.backend.set\_value
* tf.keras.backend.set\_value

tf.keras.backend.set\_value(  
    x,  
    value  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

### Used in the guide:

* [Keras custom callbacks](https://www.tensorflow.org/beta/guide/keras/custom_callback)

#### Arguments:

* **x**: Tensor to set to a new value.
* **value**: Value to set the tensor to, as a Numpy array (of the same shape).

# tf.keras.backend.shape

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/shape#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/shape#aliases)

Returns the symbolic shape of a tensor or variable.

### Aliases:

* tf.compat.v1.keras.backend.shape
* tf.compat.v2.keras.backend.shape
* tf.keras.backend.shape

tf.keras.backend.shape(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A symbolic shape (which is itself a tensor).

#### Examples:

    # TensorFlow example  
    >>> from keras import backend as K  
    >>> tf\_session = K.get\_session()  
    >>> val = np.array([[1, 2], [3, 4]])  
    >>> kvar = K.variable(value=val)  
    >>> input = keras.backend.placeholder(shape=(2, 4, 5))  
    >>> K.shape(kvar)  
    <tf.Tensor 'Shape\_8:0' shape=(2,) dtype=int32>  
    >>> K.shape(input)  
    <tf.Tensor 'Shape\_9:0' shape=(3,) dtype=int32>  
    # To get integer shape (Instead, you can use K.int\_shape(x))  
    >>> K.shape(kvar).eval(session=tf\_session)  
    array([2, 2], dtype=int32)  
    >>> K.shape(input).eval(session=tf\_session)  
    array([2, 4, 5], dtype=int32)

# tf.keras.backend.sigmoid

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sigmoid#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sigmoid#aliases)

Element-wise sigmoid.

### Aliases:

* tf.compat.v1.keras.backend.sigmoid
* tf.compat.v2.keras.backend.sigmoid
* tf.keras.backend.sigmoid

tf.keras.backend.sigmoid(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.sign

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sign#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sign#aliases)

Element-wise sign.

### Aliases:

* tf.compat.v1.keras.backend.sign
* tf.compat.v2.keras.backend.sign
* tf.keras.backend.sign

tf.keras.backend.sign(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.sin

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sin#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sin#aliases)

Computes sin of x element-wise.

### Aliases:

* tf.compat.v1.keras.backend.sin
* tf.compat.v2.keras.backend.sin
* tf.keras.backend.sin

tf.keras.backend.sin(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.softmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softmax#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softmax#aliases)

Softmax of a tensor.

### Aliases:

* tf.compat.v1.keras.backend.softmax
* tf.compat.v2.keras.backend.softmax
* tf.keras.backend.softmax

tf.keras.backend.softmax(  
    x,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.

#### Returns:

A tensor.

# tf.keras.backend.softplus

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softplus#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softplus#aliases)

Softplus of a tensor.

### Aliases:

* tf.compat.v1.keras.backend.softplus
* tf.compat.v2.keras.backend.softplus
* tf.keras.backend.softplus

tf.keras.backend.softplus(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.softsign

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softsign#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/softsign#aliases)

Softsign of a tensor.

### Aliases:

* tf.compat.v1.keras.backend.softsign
* tf.compat.v2.keras.backend.softsign
* tf.keras.backend.softsign

tf.keras.backend.softsign(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.sparse\_categorical\_crossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sparse_categorical_crossentropy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sparse_categorical_crossentropy#aliases)

Categorical crossentropy with integer targets.

### Aliases:

* tf.compat.v1.keras.backend.sparse\_categorical\_crossentropy
* tf.compat.v2.keras.backend.sparse\_categorical\_crossentropy
* tf.keras.backend.sparse\_categorical\_crossentropy

tf.keras.backend.sparse\_categorical\_crossentropy(  
    target,  
    output,  
    from\_logits=False,  
    axis=-1  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **target**: An integer tensor.
* **output**: A tensor resulting from a softmax (unless from\_logits is True, in which case output is expected to be the logits).
* **from\_logits**: Boolean, whether output is the result of a softmax, or is a tensor of logits.
* **axis**: Int specifying the channels axis. axis=-1 corresponds to data format channels\_last', andaxis=1corresponds to data formatchannels\_first`.

#### Returns:

Output tensor.

#### Raises:

* **ValueError**: if axis is neither -1 nor one of the axes of output.

# tf.keras.backend.spatial\_2d\_padding

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/spatial_2d_padding#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/spatial_2d_padding#aliases)

Pads the 2nd and 3rd dimensions of a 4D tensor.

### Aliases:

* tf.compat.v1.keras.backend.spatial\_2d\_padding
* tf.compat.v2.keras.backend.spatial\_2d\_padding
* tf.keras.backend.spatial\_2d\_padding

tf.keras.backend.spatial\_2d\_padding(  
    x,  
    padding=((1, 1), (1, 1)),  
    data\_format=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **padding**: Tuple of 2 tuples, padding pattern.
* **data\_format**: One of channels\_last or channels\_first.

#### Returns:

A padded 4D tensor.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.spatial\_3d\_padding

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/spatial_3d_padding#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/spatial_3d_padding#aliases)

Pads 5D tensor with zeros along the depth, height, width dimensions.

### Aliases:

* tf.compat.v1.keras.backend.spatial\_3d\_padding
* tf.compat.v2.keras.backend.spatial\_3d\_padding
* tf.keras.backend.spatial\_3d\_padding

tf.keras.backend.spatial\_3d\_padding(  
    x,  
    padding=((1, 1), (1, 1), (1, 1)),  
    data\_format=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Pads these dimensions with respectively "padding[0]", "padding[1]" and "padding[2]" zeros left and right.

For 'channels\_last' data\_format, the 2nd, 3rd and 4th dimension will be padded. For 'channels\_first' data\_format, the 3rd, 4th and 5th dimension will be padded.

#### Arguments:

* **x**: Tensor or variable.
* **padding**: Tuple of 3 tuples, padding pattern.
* **data\_format**: One of channels\_last or channels\_first.

#### Returns:

A padded 5D tensor.

#### Raises:

* **ValueError**: if data\_format is neither channels\_last or channels\_first.

# tf.keras.backend.sqrt

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sqrt#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sqrt#aliases)

Element-wise square root.

### Aliases:

* tf.compat.v1.keras.backend.sqrt
* tf.compat.v2.keras.backend.sqrt
* tf.keras.backend.sqrt

tf.keras.backend.sqrt(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.square

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/square#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/square#aliases)

Element-wise square.

### Aliases:

* tf.compat.v1.keras.backend.square
* tf.compat.v2.keras.backend.square
* tf.keras.backend.square

tf.keras.backend.square(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.squeeze

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/squeeze#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/squeeze#aliases)

Removes a 1-dimension from the tensor at index "axis".

### Aliases:

* tf.compat.v1.keras.backend.squeeze
* tf.compat.v2.keras.backend.squeeze
* tf.keras.backend.squeeze

tf.keras.backend.squeeze(  
    x,  
    axis  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: Axis to drop.

#### Returns:

A tensor with the same data as x but reduced dimensions.

# tf.keras.backend.stack

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/stack#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/stack#aliases)

Stacks a list of rank R tensors into a rank R+1 tensor.

### Aliases:

* tf.compat.v1.keras.backend.stack
* tf.compat.v2.keras.backend.stack
* tf.keras.backend.stack

tf.keras.backend.stack(  
    x,  
    axis=0  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: List of tensors.
* **axis**: Axis along which to perform stacking.

#### Returns:

A tensor.

# tf.keras.backend.std

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/std#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/std#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/std#used_in_the_guide)

Standard deviation of a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.std
* tf.compat.v2.keras.backend.std
* tf.keras.backend.std

tf.keras.backend.std(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to compute the standard deviation.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1. If keepdims is True, the reduced dimension is retained with length 1.

#### Returns:

A tensor with the standard deviation of elements of x.

# tf.keras.backend.stop\_gradient

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/stop_gradient#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/stop_gradient#aliases)

Returns variables but with zero gradient w.r.t. every other variable.

### Aliases:

* tf.compat.v1.keras.backend.stop\_gradient
* tf.compat.v2.keras.backend.stop\_gradient
* tf.keras.backend.stop\_gradient

tf.keras.backend.stop\_gradient(variables)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **variables**: Tensor or list of tensors to consider constant with respect to any other variable.

#### Returns:

A single tensor or a list of tensors (depending on the passed argument) that has no gradient with respect to any other variable.

# tf.keras.backend.sum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/sum#aliases)

Sum of the values in a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.sum
* tf.compat.v2.keras.backend.sum
* tf.keras.backend.sum

tf.keras.backend.sum(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to sum over.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1. If keepdims is True, the reduced dimension is retained with length 1.

#### Returns:

A tensor with sum of x.

# tf.keras.backend.switch

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/switch#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/switch#aliases)

Switches between two operations depending on a scalar value.

### Aliases:

* tf.compat.v1.keras.backend.switch
* tf.compat.v2.keras.backend.switch
* tf.keras.backend.switch

tf.keras.backend.switch(  
    condition,  
    then\_expression,  
    else\_expression  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

Note that both then\_expression and else\_expression should be symbolic tensors of the same shape.

#### Arguments:

* **condition**: tensor (int or bool).
* **then\_expression**: either a tensor, or a callable that returns a tensor.
* **else\_expression**: either a tensor, or a callable that returns a tensor.

#### Returns:

The selected tensor.

#### Raises:

* **ValueError**: If rank of condition is greater than rank of expressions.

# tf.keras.backend.tanh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/tanh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/tanh#aliases)

Element-wise tanh.

### Aliases:

* tf.compat.v1.keras.backend.tanh
* tf.compat.v2.keras.backend.tanh
* tf.keras.backend.tanh

tf.keras.backend.tanh(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.

#### Returns:

A tensor.

# tf.keras.backend.temporal\_padding

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/temporal_padding#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/temporal_padding#aliases)

Pads the middle dimension of a 3D tensor.

### Aliases:

* tf.compat.v1.keras.backend.temporal\_padding
* tf.compat.v2.keras.backend.temporal\_padding
* tf.keras.backend.temporal\_padding

tf.keras.backend.temporal\_padding(  
    x,  
    padding=(1, 1)  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.
* **padding**: Tuple of 2 integers, how many zeros to add at the start and end of dim 1.

#### Returns:

A padded 3D tensor.

# tf.keras.backend.tile

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/tile#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/tile#aliases)

Creates a tensor by tiling x by n.

### Aliases:

* tf.compat.v1.keras.backend.tile
* tf.compat.v2.keras.backend.tile
* tf.keras.backend.tile

tf.keras.backend.tile(  
    x,  
    n  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable
* **n**: A list of integer. The length must be the same as the number of dimensions in x.

#### Returns:

A tiled tensor.

# tf.keras.backend.to\_dense

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/to_dense#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/to_dense#aliases)

Converts a sparse tensor into a dense tensor and returns it.

### Aliases:

* tf.compat.v1.keras.backend.to\_dense
* tf.compat.v2.keras.backend.to\_dense
* tf.keras.backend.to\_dense

tf.keras.backend.to\_dense(tensor)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **tensor**: A tensor instance (potentially sparse).

#### Returns:

A dense tensor.

#### Examples:

    >>> from keras import backend as K  
    >>> b = K.placeholder((2, 2), sparse=True)  
    >>> print(K.is\_sparse(b))  
    True  
    >>> c = K.to\_dense(b)  
    >>> print(K.is\_sparse(c))  
    False

# tf.keras.backend.transpose

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/transpose#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/transpose#aliases)

Transposes a tensor and returns it.

### Aliases:

* tf.compat.v1.keras.backend.transpose
* tf.compat.v2.keras.backend.transpose
* tf.keras.backend.transpose

tf.keras.backend.transpose(x)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Tensor or variable.

#### Returns:

A tensor.

#### Examples:

    >>> var = K.variable([[1, 2, 3], [4, 5, 6]])  
    >>> K.eval(var)  
    array([[ 1.,  2.,  3.],  
           [ 4.,  5.,  6.]], dtype=float32)  
    >>> var\_transposed = K.transpose(var)  
    >>> K.eval(var\_transposed)  
    array([[ 1.,  4.],  
           [ 2.,  5.],  
           [ 3.,  6.]], dtype=float32)

    >>> input = K.placeholder((2, 3))  
    >>> input  
    <tf.Tensor 'Placeholder\_11:0' shape=(2, 3) dtype=float32>  
    >>> input\_transposed = K.transpose(input)  
    >>> input\_transposed  
    <tf.Tensor 'transpose\_4:0' shape=(3, 2) dtype=float32>

# tf.keras.backend.truncated\_normal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/truncated_normal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/truncated_normal#aliases)

Returns a tensor with truncated random normal distribution of values.

### Aliases:

* tf.compat.v1.keras.backend.truncated\_normal
* tf.compat.v2.keras.backend.truncated\_normal
* tf.keras.backend.truncated\_normal

tf.keras.backend.truncated\_normal(  
    shape,  
    mean=0.0,  
    stddev=1.0,  
    dtype=None,  
    seed=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

The generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than two standard deviations from the mean are dropped and re-picked.

#### Arguments:

* **shape**: A tuple of integers, the shape of tensor to create.
* **mean**: Mean of the values.
* **stddev**: Standard deviation of the values.
* **dtype**: String, dtype of returned tensor.
* **seed**: Integer, random seed.

#### Returns:

A tensor.

tf.keras.backend.update

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update#aliases)

Aliases:

* tf.compat.v1.keras.backend.update
* tf.compat.v2.keras.backend.update
* tf.keras.backend.update

tf.keras.backend.update(  
    x,  
    new\_x  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

# tf.keras.backend.update\_add

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update_add#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update_add#aliases)

Update the value of x by adding increment.

### Aliases:

* tf.compat.v1.keras.backend.update\_add
* tf.compat.v2.keras.backend.update\_add
* tf.keras.backend.update\_add

tf.keras.backend.update\_add(  
    x,  
    increment  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A Variable.
* **increment**: A tensor of same shape as x.

#### Returns:

The variable x updated.

# tf.keras.backend.update\_sub

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update_sub#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/update_sub#aliases)

Update the value of x by subtracting decrement.

### Aliases:

* tf.compat.v1.keras.backend.update\_sub
* tf.compat.v2.keras.backend.update\_sub
* tf.keras.backend.update\_sub

tf.keras.backend.update\_sub(  
    x,  
    decrement  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A Variable.
* **decrement**: A tensor of same shape as x.

#### Returns:

The variable x updated.

# tf.keras.backend.var

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/var#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/var#aliases)

Variance of a tensor, alongside the specified axis.

### Aliases:

* tf.compat.v1.keras.backend.var
* tf.compat.v2.keras.backend.var
* tf.keras.backend.var

tf.keras.backend.var(  
    x,  
    axis=None,  
    keepdims=False  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: A tensor or variable.
* **axis**: An integer, the axis to compute the variance.
* **keepdims**: A boolean, whether to keep the dimensions or not. If keepdims is False, the rank of the tensor is reduced by 1. If keepdims is True, the reduced dimension is retained with length 1.

#### Returns:

A tensor with the variance of elements of x.

# tf.keras.backend.variable

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/variable#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/variable#aliases)

Instantiates a variable and returns it.

### Aliases:

* tf.compat.v1.keras.backend.variable
* tf.compat.v2.keras.backend.variable
* tf.keras.backend.variable

tf.keras.backend.variable(  
    value,  
    dtype=None,  
    name=None,  
    constraint=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **value**: Numpy array, initial value of the tensor.
* **dtype**: Tensor type.
* **name**: Optional name string for the tensor.
* **constraint**: Optional projection function to be applied to the variable after an optimizer update.

#### Returns:

A variable instance (with Keras metadata included).

#### Examples:

    >>> import numpy as np  
    >>> from keras import backend as K  
    >>> val = np.array([[1, 2], [3, 4]])  
    >>> kvar = K.variable(value=val, dtype='float64', name='example\_var')  
    >>> K.dtype(kvar)  
    'float64'  
    >>> print(kvar)  
    example\_var  
    >>> kvar.eval()  
    array([[ 1.,  2.],  
           [ 3.,  4.]])

# tf.keras.backend.zeros

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/zeros#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/zeros#aliases)

Instantiates an all-zeros variable and returns it.

### Aliases:

* tf.compat.v1.keras.backend.zeros
* tf.compat.v2.keras.backend.zeros
* tf.keras.backend.zeros

tf.keras.backend.zeros(  
    shape,  
    dtype=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **shape**: Tuple of integers, shape of returned Keras variable
* **dtype**: String, data type of returned Keras variable
* **name**: String, name of returned Keras variable

#### Returns:

A variable (including Keras metadata), filled with 0.0. Note that if shape was symbolic, we cannot return a variable, and will return a dynamically-shaped tensor instead.

#### Example:

    >>> from keras import backend as K  
    >>> kvar = K.zeros((3,4))  
    >>> K.eval(kvar)  
    array([[ 0.,  0.,  0.,  0.],  
           [ 0.,  0.,  0.,  0.],  
           [ 0.,  0.,  0.,  0.]], dtype=float32)

# tf.keras.backend.zeros\_like

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/zeros_like#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/backend/zeros_like#aliases)

Instantiates an all-zeros variable of the same shape as another tensor.

### Aliases:

* tf.compat.v1.keras.backend.zeros\_like
* tf.compat.v2.keras.backend.zeros\_like
* tf.keras.backend.zeros\_like

tf.keras.backend.zeros\_like(  
    x,  
    dtype=None,  
    name=None  
)

Defined in [python/keras/backend.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/backend.py).

#### Arguments:

* **x**: Keras variable or Keras tensor.
* **dtype**: String, dtype of returned Keras variable. None uses the dtype of x.
* **name**: String, name for the variable to create.

#### Returns:

A Keras variable with the shape of x filled with zeros.

#### Example:

    >>> from keras import backend as K  
    >>> kvar = K.variable(np.random.random((2,3)))  
    >>> kvar\_zeros = K.zeros\_like(kvar)  
    >>> K.eval(kvar\_zeros)  
    array([[ 0.,  0.,  0.],  
           [ 0.,  0.,  0.]], dtype=float32)

Module: tf.keras.callbacks

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks#classes)

Callbacks: utilities called at certain points during model training.

Aliases:

* Module tf.compat.v2.keras.callbacks
* Module tf.keras.callbacks

Defined in [python/keras/api/\_v2/keras/callbacks/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/callbacks/__init__.py).

Classes

[class BaseLogger](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/BaseLogger): Callback that accumulates epoch averages of metrics.

[class CSVLogger](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/CSVLogger): Callback that streams epoch results to a csv file.

[class Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback): Abstract base class used to build new callbacks.

[class EarlyStopping](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/EarlyStopping): Stop training when a monitored quantity has stopped improving.

[class History](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/History): Callback that records events into a History object.

[class LambdaCallback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LambdaCallback): Callback for creating simple, custom callbacks on-the-fly.

[class LearningRateScheduler](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LearningRateScheduler): Learning rate scheduler.

[class ModelCheckpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ModelCheckpoint): Save the model after every epoch.

[class ProgbarLogger](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ProgbarLogger): Callback that prints metrics to stdout.

[class ReduceLROnPlateau](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau): Reduce learning rate when a metric has stopped improving.

[class RemoteMonitor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/RemoteMonitor): Callback used to stream events to a server.

[class TensorBoard](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TensorBoard): Enable visualizations for TensorBoard.

[class TerminateOnNaN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TerminateOnNaN): Callback that terminates training when a NaN loss is encountered.

# tf.keras.callbacks.BaseLogger

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/BaseLogger#top_of_page)
* [Class BaseLogger](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/BaseLogger#class_baselogger)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/BaseLogger#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/BaseLogger#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/BaseLogger#methods)

## Class BaseLogger

Callback that accumulates epoch averages of metrics.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.BaseLogger
* Class tf.compat.v2.keras.callbacks.BaseLogger
* Class tf.keras.callbacks.BaseLogger

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

This callback is automatically applied to every Keras model.

#### Arguments:

* **stateful\_metrics**: Iterable of string names of metrics that should not be averaged over an epoch. Metrics in this list will be logged as-is in on\_epoch\_end. All others will be averaged in on\_epoch\_end.

## \_\_init\_\_

\_\_init\_\_(stateful\_metrics=None)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.Callback

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback#top_of_page)
* [Class Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback#class_callback)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback#methods)

## Class Callback

Abstract base class used to build new callbacks.

### Aliases:

* Class tf.compat.v1.keras.callbacks.Callback
* Class tf.compat.v2.keras.callbacks.Callback
* Class tf.keras.callbacks.Callback

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

#### Attributes:

* **params**: dict. Training parameters (eg. verbosity, batch size, number of epochs...).
* **model**: instance of keras.models.Model. Reference of the model being trained.
* **validation\_data**: Deprecated. Do not use.

The logs dictionary that callback methods take as argument will contain keys for quantities relevant to the current batch or epoch.

Currently, the .fit() method of the Model class will include the following quantities in the logsthat it passes to its callbacks:

on\_epoch\_end: logs include `acc` and `loss`, and  
    optionally include `val\_loss`  
    (if validation is enabled in `fit`), and `val\_acc`  
    (if validation and accuracy monitoring are enabled).  
on\_batch\_begin: logs include `size`,  
    the number of samples in the current batch.  
on\_batch\_end: logs include `loss`, and optionally `acc`  
    (if accuracy monitoring is enabled).

## \_\_init\_\_

\_\_init\_\_()

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

Called at the end of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict, metric results for this training epoch, and for the validation epoch if validation is performed. Validation result keys are prefixed with val\_.

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.CSVLogger

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/CSVLogger#top_of_page)
* [Class CSVLogger](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/CSVLogger#class_csvlogger)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/CSVLogger#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/CSVLogger#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/CSVLogger#methods)

## Class CSVLogger

Callback that streams epoch results to a csv file.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.CSVLogger
* Class tf.compat.v2.keras.callbacks.CSVLogger
* Class tf.keras.callbacks.CSVLogger

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

Supports all values that can be represented as a string, including 1D iterables such as np.ndarray.

#### Example:

csv\_logger = CSVLogger('training.log')  
model.fit(X\_train, Y\_train, callbacks=[csv\_logger])

#### Arguments:

* **filename**: filename of the csv file, e.g. 'run/log.csv'.
* **separator**: string used to separate elements in the csv file.
* **append**: True: append if file exists (useful for continuing training). False: overwrite existing file,

## \_\_init\_\_

\_\_init\_\_(  
    filename,  
    separator=',',  
    append=False  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.EarlyStopping

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/EarlyStopping#top_of_page)
* [Class EarlyStopping](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/EarlyStopping#class_earlystopping)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/EarlyStopping#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/EarlyStopping#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/EarlyStopping#used_in_the_tutorials)

## Class EarlyStopping

Stop training when a monitored quantity has stopped improving.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.EarlyStopping
* Class tf.compat.v2.keras.callbacks.EarlyStopping
* Class tf.keras.callbacks.EarlyStopping

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Regression: Predict fuel efficiency](https://www.tensorflow.org/beta/tutorials/keras/basic_regression)

#### Arguments:

* **monitor**: Quantity to be monitored.
* **min\_delta**: Minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min\_delta, will count as no improvement.
* **patience**: Number of epochs with no improvement after which training will be stopped.
* **verbose**: verbosity mode.
* **mode**: One of {"auto", "min", "max"}. In min mode, training will stop when the quantity monitored has stopped decreasing; in max mode it will stop when the quantity monitored has stopped increasing; in auto mode, the direction is automatically inferred from the name of the monitored quantity.
* **baseline**: Baseline value for the monitored quantity. Training will stop if the model doesn't show improvement over the baseline.
* **restore\_best\_weights**: Whether to restore model weights from the epoch with the best value of the monitored quantity. If False, the model weights obtained at the last step of training are used.

#### Example:

callback = tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=3)  
# This callback will stop the training when there is no improvement in  
# the validation loss for three consecutive epochs.  
model.fit(data, labels, epochs=100, callbacks=[callback],  
    validation\_data=(val\_data, val\_labels))

## \_\_init\_\_

\_\_init\_\_(  
    monitor='val\_loss',  
    min\_delta=0,  
    patience=0,  
    verbose=0,  
    mode='auto',  
    baseline=None,  
    restore\_best\_weights=False  
)

## Methods

### get\_monitor\_value

get\_monitor\_value(logs)

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.History

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/History#top_of_page)
* [Class History](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/History#class_history)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/History#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/History#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/History#methods)

## Class History

Callback that records events into a History object.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.History
* Class tf.compat.v2.keras.callbacks.History
* Class tf.keras.callbacks.History

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

This callback is automatically applied to every Keras model. The History object gets returned by the fit method of models.

## \_\_init\_\_

\_\_init\_\_()

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.LambdaCallback

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LambdaCallback#top_of_page)
* [Class LambdaCallback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LambdaCallback#class_lambdacallback)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LambdaCallback#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LambdaCallback#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LambdaCallback#methods)

## Class LambdaCallback

Callback for creating simple, custom callbacks on-the-fly.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.LambdaCallback
* Class tf.compat.v2.keras.callbacks.LambdaCallback
* Class tf.keras.callbacks.LambdaCallback

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

This callback is constructed with anonymous functions that will be called at the appropriate time. Note that the callbacks expects positional **Arguments**, as:

* on\_epoch\_begin and on\_epoch\_end expect two positional **Arguments**: epoch, logs
* on\_batch\_begin and on\_batch\_end expect two positional **Arguments**: batch, logs
* on\_train\_begin and on\_train\_end expect one positional argument: logs

#### Arguments:

* **on\_epoch\_begin**: called at the beginning of every epoch.
* **on\_epoch\_end**: called at the end of every epoch.
* **on\_batch\_begin**: called at the beginning of every batch.
* **on\_batch\_end**: called at the end of every batch.
* **on\_train\_begin**: called at the beginning of model training.
* **on\_train\_end**: called at the end of model training.

#### Example:

# Print the batch number at the beginning of every batch.  
batch\_print\_callback = LambdaCallback(  
    on\_batch\_begin=lambda batch,logs: print(batch))  
  
# Stream the epoch loss to a file in JSON format. The file content  
# is not well-formed JSON but rather has a JSON object per line.  
import json  
json\_log = open('loss\_log.json', mode='wt', buffering=1)  
json\_logging\_callback = LambdaCallback(  
    on\_epoch\_end=lambda epoch, logs: json\_log.write(  
        json.dumps({'epoch': epoch, 'loss': logs['loss']}) + '\n'),  
    on\_train\_end=lambda logs: json\_log.close()  
)  
  
# Terminate some processes after having finished model training.  
processes = ...  
cleanup\_callback = LambdaCallback(  
    on\_train\_end=lambda logs: [  
        p.terminate() for p in processes if p.is\_alive()])  
  
model.fit(...,  
          callbacks=[batch\_print\_callback,  
                     json\_logging\_callback,  
                     cleanup\_callback])

## \_\_init\_\_

\_\_init\_\_(  
    on\_epoch\_begin=None,  
    on\_epoch\_end=None,  
    on\_batch\_begin=None,  
    on\_batch\_end=None,  
    on\_train\_begin=None,  
    on\_train\_end=None,  
    \*\*kwargs  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

Called at the end of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict, metric results for this training epoch, and for the validation epoch if validation is performed. Validation result keys are prefixed with val\_.

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.LearningRateScheduler

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LearningRateScheduler#top_of_page)
* [Class LearningRateScheduler](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LearningRateScheduler#class_learningratescheduler)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LearningRateScheduler#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LearningRateScheduler#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/LearningRateScheduler#__init__)

## Class LearningRateScheduler

Learning rate scheduler.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.LearningRateScheduler
* Class tf.compat.v2.keras.callbacks.LearningRateScheduler
* Class tf.keras.callbacks.LearningRateScheduler

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

### Used in the tutorials:

* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)

#### Arguments:

* **schedule**: a function that takes an epoch index as input (integer, indexed from 0) and returns a new learning rate as output (float).
* **verbose**: int. 0: quiet, 1: update messages.

# This function keeps the learning rate at 0.001 for the first ten epochs  
# and decreases it exponentially after that.  
def scheduler(epoch):  
  if epoch < 10:  
    return 0.001  
  else:  
    return 0.001 \* tf.math.exp(0.1 \* (10 - epoch))  
  
callback = tf.keras.callbacks.LearningRateScheduler(scheduler)  
model.fit(data, labels, epochs=100, callbacks=[callback],  
          validation\_data=(val\_data, val\_labels))

## \_\_init\_\_

\_\_init\_\_(  
    schedule,  
    verbose=0  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.ModelCheckpoint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ModelCheckpoint#top_of_page)
* [Class ModelCheckpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ModelCheckpoint#class_modelcheckpoint)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ModelCheckpoint#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ModelCheckpoint#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ModelCheckpoint#used_in_the_tutorials)

## Class ModelCheckpoint

Save the model after every epoch.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.ModelCheckpoint
* Class tf.compat.v2.keras.callbacks.ModelCheckpoint
* Class tf.keras.callbacks.ModelCheckpoint

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)

filepath can contain named formatting options, which will be filled the value of epoch and keys in logs (passed in on\_epoch\_end).

For example: if filepath is weights.{epoch:02d}-{val\_loss:.2f}.hdf5, then the model checkpoints will be saved with the epoch number and the validation loss in the filename.

#### Arguments:

* **filepath**: string, path to save the model file.
* **monitor**: quantity to monitor.
* **verbose**: verbosity mode, 0 or 1.
* **save\_best\_only**: if save\_best\_only=True, the latest best model according to the quantity monitored will not be overwritten.
* **mode**: one of {auto, min, max}. If save\_best\_only=True, the decision to overwrite the current save file is made based on either the maximization or the minimization of the monitored quantity. For val\_acc, this should be max, for val\_loss this should be min, etc. In auto mode, the direction is automatically inferred from the name of the monitored quantity.
* **save\_weights\_only**: if True, then only the model's weights will be saved (model.save\_weights(filepath)), else the full model is saved (model.save(filepath)).
* **save\_freq**: 'epoch' or integer. When using 'epoch', the callback saves the model after each epoch. When using integer, the callback saves the model at end of a batch at which this many samples have been seen since last saving. Note that if the saving isn't aligned to epochs, the monitored metric may potentially be less reliable (it could reflect as little as 1 batch, since the metrics get reset every epoch). Defaults to 'epoch'
* **load\_weights\_on\_restart**: Whether the training should restore the model. If True, the model will attempt to load the checkpoint file from filepath at the start of model.fit(). This saves the need of manually calling model.load\_weights() before `model.fit(). In multi-worker distributed training, this provides fault-tolerance and loads the model automatically upon recovery of workers. The callback gives up loading if the filepath does not exist, and raises ValueError if format does not match. Defaults to False.
* **\*\*kwargs**: Additional **Arguments** for backwards compatibility. Possible key is period.

## \_\_init\_\_

\_\_init\_\_(  
    filepath,  
    monitor='val\_loss',  
    verbose=0,  
    save\_best\_only=False,  
    save\_weights\_only=False,  
    mode='auto',  
    save\_freq='epoch',  
    load\_weights\_on\_restart=False,  
    \*\*kwargs  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.ProgbarLogger

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ProgbarLogger#top_of_page)
* [Class ProgbarLogger](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ProgbarLogger#class_progbarlogger)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ProgbarLogger#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ProgbarLogger#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ProgbarLogger#methods)

## Class ProgbarLogger

Callback that prints metrics to stdout.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.ProgbarLogger
* Class tf.compat.v2.keras.callbacks.ProgbarLogger
* Class tf.keras.callbacks.ProgbarLogger

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

#### Arguments:

* **count\_mode**: One of "steps" or "samples". Whether the progress bar should count samples seen or steps (batches) seen.
* **stateful\_metrics**: Iterable of string names of metrics that should not be averaged over an epoch. Metrics in this list will be logged as-is. All others will be averaged over time (e.g. loss, etc).

#### Raises:

* **ValueError**: In case of invalid count\_mode.

## \_\_init\_\_

\_\_init\_\_(  
    count\_mode='samples',  
    stateful\_metrics=None  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.ReduceLROnPlateau

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau#top_of_page)
* [Class ReduceLROnPlateau](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau#class_reducelronplateau)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau#methods)

## Class ReduceLROnPlateau

Reduce learning rate when a metric has stopped improving.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.ReduceLROnPlateau
* Class tf.compat.v2.keras.callbacks.ReduceLROnPlateau
* Class tf.keras.callbacks.ReduceLROnPlateau

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

#### Example:

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.2,  
                              patience=5, min\_lr=0.001)  
model.fit(X\_train, Y\_train, callbacks=[reduce\_lr])

#### Arguments:

* **monitor**: quantity to be monitored.
* **factor**: factor by which the learning rate will be reduced. new\_lr = lr \* factor
* **patience**: number of epochs with no improvement after which learning rate will be reduced.
* **verbose**: int. 0: quiet, 1: update messages.
* **mode**: one of {auto, min, max}. In min mode, lr will be reduced when the quantity monitored has stopped decreasing; in max mode it will be reduced when the quantity monitored has stopped increasing; in auto mode, the direction is automatically inferred from the name of the monitored quantity.
* **min\_delta**: threshold for measuring the new optimum, to only focus on significant changes.
* **cooldown**: number of epochs to wait before resuming normal operation after lr has been reduced.
* **min\_lr**: lower bound on the learning rate.

## \_\_init\_\_

\_\_init\_\_(  
    monitor='val\_loss',  
    factor=0.1,  
    patience=10,  
    verbose=0,  
    mode='auto',  
    min\_delta=0.0001,  
    cooldown=0,  
    min\_lr=0,  
    \*\*kwargs  
)

## Methods

### in\_cooldown

in\_cooldown()

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.RemoteMonitor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/RemoteMonitor#top_of_page)
* [Class RemoteMonitor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/RemoteMonitor#class_remotemonitor)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/RemoteMonitor#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/RemoteMonitor#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/RemoteMonitor#methods)

## Class RemoteMonitor

Callback used to stream events to a server.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.RemoteMonitor
* Class tf.compat.v2.keras.callbacks.RemoteMonitor
* Class tf.keras.callbacks.RemoteMonitor

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

Requires the requests library. Events are sent to root + '/publish/epoch/end/' by default. Calls are HTTP POST, with a data argument which is a JSON-encoded dictionary of event data. If send\_as\_json is set to True, the content type of the request will be application/json. Otherwise the serialized JSON will be sent within a form.

#### Arguments:

* **root**: String; root url of the target server.
* **path**: String; path relative to root to which the events will be sent.
* **field**: String; JSON field under which the data will be stored. The field is used only if the payload is sent within a form (i.e. send\_as\_json is set to False).
* **headers**: Dictionary; optional custom HTTP headers.
* **send\_as\_json**: Boolean; whether the request should be sent as application/json.

## \_\_init\_\_

\_\_init\_\_(  
    root='http://localhost:9000',  
    path='/publish/epoch/end/',  
    field='data',  
    headers=None,  
    send\_as\_json=False  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_end.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

# tf.keras.callbacks.TensorBoard

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TensorBoard#top_of_page)
* [Class TensorBoard](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TensorBoard#class_tensorboard)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TensorBoard#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TensorBoard#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TensorBoard#used_in_the_tutorials)

## Class TensorBoard

Enable visualizations for TensorBoard.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v2.keras.callbacks.TensorBoard
* Class tf.keras.callbacks.TensorBoard

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)

### Used in the tutorials:

* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)

TensorBoard is a visualization tool provided with TensorFlow.

This callback logs events for TensorBoard, including: \* Metrics summary plots \* Training graph visualization \* Activation histograms \* Sampled profiling

If you have installed TensorFlow with pip, you should be able to launch TensorBoard from the command line:

tensorboard --logdir=path\_to\_your\_logs

You can find more information about TensorBoard [here](https://www.tensorflow.org/get_started/summaries_and_tensorboard).

#### Arguments:

* **log\_dir**: the path of the directory where to save the log files to be parsed by TensorBoard.
* **histogram\_freq**: frequency (in epochs) at which to compute activation and weight histograms for the layers of the model. If set to 0, histograms won't be computed. Validation data (or split) must be specified for histogram visualizations.
* **write\_graph**: whether to visualize the graph in TensorBoard. The log file can become quite large when write\_graph is set to True.
* **write\_images**: whether to write model weights to visualize as image in TensorBoard.
* **update\_freq**: 'batch' or 'epoch' or integer. When using 'batch', writes the losses and metrics to TensorBoard after each batch. The same applies for 'epoch'. If using an integer, let's say 1000, the callback will write the metrics and losses to TensorBoard every 1000 samples. Note that writing too frequently to TensorBoard can slow down your training.
* **profile\_batch**: Profile the batch to sample compute characteristics. By default, it will profile the second batch. Set profile\_batch=0 to disable profiling. Must run in TensorFlow eager mode.
* **embeddings\_freq**: frequency (in epochs) at which embedding layers will be visualized. If set to 0, embeddings won't be visualized.
* **embeddings\_metadata**: a dictionary which maps layer name to a file name in which metadata for this embedding layer is saved. See the [details](https://www.tensorflow.org/how_tos/embedding_viz/#metadata_optional) about metadata files format. In case if the same metadata file is used for all embedding layers, string can be passed.

#### Raises:

* **ValueError**: If histogram\_freq is set and no validation data is provided.

## \_\_init\_\_

\_\_init\_\_(  
    log\_dir='logs',  
    histogram\_freq=0,  
    write\_graph=True,  
    write\_images=False,  
    update\_freq='epoch',  
    profile\_batch=2,  
    embeddings\_freq=0,  
    embeddings\_metadata=None,  
    \*\*kwargs  
)

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

Writes scalar summaries for metrics on every training batch.

Performs profiling if current batch is in profiler\_batches.

#### Arguments:

* **batch**: Integer, index of batch within the current epoch.
* **logs**: Dict. Metric results for this batch.

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

Runs metrics and histogram summaries at epoch end.

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

### on\_train\_end

on\_train\_end(logs=None)

### set\_model

set\_model(model)

Sets Keras model and writes graph if specified.

### set\_params

set\_params(params)

# tf.keras.callbacks.TerminateOnNaN

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TerminateOnNaN#top_of_page)
* [Class TerminateOnNaN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TerminateOnNaN#class_terminateonnan)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TerminateOnNaN#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TerminateOnNaN#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/TerminateOnNaN#methods)

## Class TerminateOnNaN

Callback that terminates training when a NaN loss is encountered.

Inherits From: [Callback](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/callbacks/Callback)

### Aliases:

* Class tf.compat.v1.keras.callbacks.TerminateOnNaN
* Class tf.compat.v2.keras.callbacks.TerminateOnNaN
* Class tf.keras.callbacks.TerminateOnNaN

Defined in [python/keras/callbacks.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/callbacks.py).

## \_\_init\_\_

\_\_init\_\_()

## Methods

### on\_batch\_begin

on\_batch\_begin(  
    batch,  
    logs=None  
)

A backwards compatibility alias for on\_train\_batch\_begin.

### on\_batch\_end

on\_batch\_end(  
    batch,  
    logs=None  
)

### on\_epoch\_begin

on\_epoch\_begin(  
    epoch,  
    logs=None  
)

Called at the start of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_epoch\_end

on\_epoch\_end(  
    epoch,  
    logs=None  
)

Called at the end of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

#### Arguments:

* **epoch**: integer, index of epoch.
* **logs**: dict, metric results for this training epoch, and for the validation epoch if validation is performed. Validation result keys are prefixed with val\_.

### on\_predict\_batch\_begin

on\_predict\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_predict\_batch\_end

on\_predict\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in predict methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_predict\_begin

on\_predict\_begin(logs=None)

Called at the beginning of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_predict\_end

on\_predict\_end(logs=None)

Called at the end of prediction.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_batch\_begin

on\_test\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a batch in evaluate methods.

Also called at the beginning of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_test\_batch\_end

on\_test\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a batch in evaluate methods.

Also called at the end of a validation batch in the fit methods, if validation data is provided.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_test\_begin

on\_test\_begin(logs=None)

Called at the beginning of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_test\_end

on\_test\_end(logs=None)

Called at the end of evaluation or validation.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_batch\_begin

on\_train\_batch\_begin(  
    batch,  
    logs=None  
)

Called at the beginning of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Has keys batch and size representing the current batch number and the size of the batch.

### on\_train\_batch\_end

on\_train\_batch\_end(  
    batch,  
    logs=None  
)

Called at the end of a training batch in fit methods.

Subclasses should override for any actions to run.

#### Arguments:

* **batch**: integer, index of batch within the current epoch.
* **logs**: dict. Metric results for this batch.

### on\_train\_begin

on\_train\_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### on\_train\_end

on\_train\_end(logs=None)

Called at the end of training.

Subclasses should override for any actions to run.

#### Arguments:

* **logs**: dict. Currently no data is passed to this argument for this method but that may change in the future.

### set\_model

set\_model(model)

### set\_params

set\_params(params)

Module: tf.keras.constraints

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints#functions)

Constraints: functions that impose constraints on weight values.

Aliases:

* Module tf.compat.v2.keras.constraints
* Module tf.keras.constraints

Defined in [python/keras/api/\_v2/keras/constraints/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/constraints/__init__.py).

Classes

[class Constraint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint)

[class MaxNorm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm): MaxNorm weight constraint.

[class MinMaxNorm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm): MinMaxNorm weight constraint.

[class NonNeg](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg): Constrains the weights to be non-negative.

[class UnitNorm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm): Constrains the weights incident to each hidden unit to have unit norm.

[class max\_norm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm): MaxNorm weight constraint.

[class min\_max\_norm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm): MinMaxNorm weight constraint.

[class non\_neg](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg): Constrains the weights to be non-negative.

[class unit\_norm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm): Constrains the weights incident to each hidden unit to have unit norm.

Functions

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/deserialize)

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/get)

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/serialize)

tf.keras.constraints.Constraint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint#top_of_page)
* [Class Constraint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint#class_constraint)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint#__call__)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint#get_config)

Class Constraint

Aliases:

* Class tf.compat.v1.keras.constraints.Constraint
* Class tf.compat.v2.keras.constraints.Constraint
* Class tf.keras.constraints.Constraint

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

Methods

\_\_call\_\_

\_\_call\_\_(w)

get\_config

get\_config()

tf.keras.constraints.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/deserialize#aliases)

Aliases:

* tf.compat.v1.keras.constraints.deserialize
* tf.compat.v2.keras.constraints.deserialize
* tf.keras.constraints.deserialize

tf.keras.constraints.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

tf.keras.constraints.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/get#aliases)

Aliases:

* tf.compat.v1.keras.constraints.get
* tf.compat.v2.keras.constraints.get
* tf.keras.constraints.get

tf.keras.constraints.get(identifier)

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

# tf.keras.constraints.MaxNorm

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#top_of_page)
* [Class MaxNorm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#class_maxnorm)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#__call__)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MaxNorm#get_config)

## Class MaxNorm

MaxNorm weight constraint.

Inherits From: [Constraint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint)

### Aliases:

* Class tf.compat.v1.keras.constraints.MaxNorm
* Class tf.compat.v1.keras.constraints.max\_norm
* Class tf.compat.v2.keras.constraints.MaxNorm
* Class tf.compat.v2.keras.constraints.max\_norm
* Class tf.keras.constraints.MaxNorm
* Class tf.keras.constraints.max\_norm

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

Constrains the weights incident to each hidden unit to have a norm less than or equal to a desired value.

#### Arguments:

* **m**: the maximum norm for the incoming weights.
* **axis**: integer, axis along which to calculate weight norms. For instance, in a Dense layer the weight matrix has shape (input\_dim, output\_dim), set axis to 0 to constrain each weight vector of length (input\_dim,). In a Conv2D layer with data\_format="channels\_last", the weight tensor has shape (rows, cols, input\_depth, output\_depth), set axis to [0, 1, 2] to constrain the weights of each filter tensor of size (rows, cols, input\_depth).

## \_\_init\_\_

\_\_init\_\_(  
    max\_value=2,  
    axis=0  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(w)

### get\_config

get\_config()

# tf.keras.constraints.MinMaxNorm

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#top_of_page)
* [Class MinMaxNorm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#class_minmaxnorm)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#__call__)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/MinMaxNorm#get_config)

## Class MinMaxNorm

MinMaxNorm weight constraint.

Inherits From: [Constraint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint)

### Aliases:

* Class tf.compat.v1.keras.constraints.MinMaxNorm
* Class tf.compat.v1.keras.constraints.min\_max\_norm
* Class tf.compat.v2.keras.constraints.MinMaxNorm
* Class tf.compat.v2.keras.constraints.min\_max\_norm
* Class tf.keras.constraints.MinMaxNorm
* Class tf.keras.constraints.min\_max\_norm

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

Constrains the weights incident to each hidden unit to have the norm between a lower bound and an upper bound.

#### Arguments:

* **min\_value**: the minimum norm for the incoming weights.
* **max\_value**: the maximum norm for the incoming weights.
* **rate**: rate for enforcing the constraint: weights will be rescaled to yield (1 - rate) \* norm + rate \* norm.clip(min\_value, max\_value). Effectively, this means that rate=1.0 stands for strict enforcement of the constraint, while rate<1.0 means that weights will be rescaled at each step to slowly move towards a value inside the desired interval.
* **axis**: integer, axis along which to calculate weight norms. For instance, in a Dense layer the weight matrix has shape (input\_dim, output\_dim), set axis to 0 to constrain each weight vector of length (input\_dim,). In a Conv2D layer with data\_format="channels\_last", the weight tensor has shape (rows, cols, input\_depth, output\_depth), set axis to [0, 1, 2] to constrain the weights of each filter tensor of size (rows, cols, input\_depth).

## \_\_init\_\_

\_\_init\_\_(  
    min\_value=0.0,  
    max\_value=1.0,  
    rate=1.0,  
    axis=0  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(w)

### get\_config

get\_config()

tf.keras.constraints.NonNeg

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg#top_of_page)
* [Class NonNeg](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg#class_nonneg)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg#__call__)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/NonNeg#get_config)

Class NonNeg

Constrains the weights to be non-negative.

Inherits From: [Constraint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint)

Aliases:

* Class tf.compat.v1.keras.constraints.NonNeg
* Class tf.compat.v1.keras.constraints.non\_neg
* Class tf.compat.v2.keras.constraints.NonNeg
* Class tf.compat.v2.keras.constraints.non\_neg
* Class tf.keras.constraints.NonNeg
* Class tf.keras.constraints.non\_neg

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

Methods

\_\_call\_\_

\_\_call\_\_(w)

get\_config

get\_config()

tf.keras.constraints.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/serialize#aliases)

Aliases:

* tf.compat.v1.keras.constraints.serialize
* tf.compat.v2.keras.constraints.serialize
* tf.keras.constraints.serialize

tf.keras.constraints.serialize(constraint)

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

# tf.keras.constraints.UnitNorm

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#top_of_page)
* [Class UnitNorm](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#class_unitnorm)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#__call__)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/UnitNorm#get_config)

## Class UnitNorm

Constrains the weights incident to each hidden unit to have unit norm.

Inherits From: [Constraint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/constraints/Constraint)

### Aliases:

* Class tf.compat.v1.keras.constraints.UnitNorm
* Class tf.compat.v1.keras.constraints.unit\_norm
* Class tf.compat.v2.keras.constraints.UnitNorm
* Class tf.compat.v2.keras.constraints.unit\_norm
* Class tf.keras.constraints.UnitNorm
* Class tf.keras.constraints.unit\_norm

Defined in [python/keras/constraints.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/constraints.py).

#### Arguments:

* **axis**: integer, axis along which to calculate weight norms. For instance, in a Dense layer the weight matrix has shape (input\_dim, output\_dim), set axis to 0 to constrain each weight vector of length (input\_dim,). In a Conv2D layer with data\_format="channels\_last", the weight tensor has shape (rows, cols, input\_depth, output\_depth), set axis to [0, 1, 2] to constrain the weights of each filter tensor of size (rows, cols, input\_depth).

## \_\_init\_\_

\_\_init\_\_(axis=0)

## Methods

### \_\_call\_\_

\_\_call\_\_(w)

### get\_config

get\_config()

Module: tf.keras.datasets

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets#aliases)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets#modules)

Keras built-in datasets.

Aliases:

* Module tf.compat.v2.keras.datasets
* Module tf.keras.datasets

Defined in [python/keras/api/\_v2/keras/datasets/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/datasets/__init__.py).

Modules

[boston\_housing](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/boston_housing) module: Boston housing price regression dataset.

[cifar10](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar10) module: CIFAR10 small images classification dataset.

[cifar100](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar100) module: CIFAR100 small images classification dataset.

[fashion\_mnist](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/fashion_mnist) module: Fashion-MNIST dataset.

[imdb](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/imdb) module: IMDB sentiment classification dataset.

[mnist](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/mnist) module: MNIST handwritten digits dataset.

[reuters](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/reuters) module: Reuters topic classification dataset.

# tf.keras.datasets.boston\_housing.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/boston_housing/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/boston_housing/load_data#aliases)

Loads the Boston Housing dataset.

### Aliases:

* tf.compat.v1.keras.datasets.boston\_housing.load\_data
* tf.compat.v2.keras.datasets.boston\_housing.load\_data
* tf.keras.datasets.boston\_housing.load\_data

tf.keras.datasets.boston\_housing.load\_data(  
    path='boston\_housing.npz',  
    test\_split=0.2,  
    seed=113  
)

Defined in [python/keras/datasets/boston\_housing.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/boston_housing.py).

#### Arguments:

* **path**: path where to cache the dataset locally (relative to ~/.keras/datasets).
* **test\_split**: fraction of the data to reserve as test set.
* **seed**: Random seed for shuffling the data before computing the test split.

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

# tf.keras.datasets.cifar10.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar10/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar10/load_data#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar10/load_data#used_in_the_guide)

Loads CIFAR10 dataset.

### Aliases:

* tf.compat.v1.keras.datasets.cifar10.load\_data
* tf.compat.v2.keras.datasets.cifar10.load\_data
* tf.keras.datasets.cifar10.load\_data

tf.keras.datasets.cifar10.load\_data()

Defined in [python/keras/datasets/cifar10.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/cifar10.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

# tf.keras.datasets.cifar100.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar100/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/cifar100/load_data#aliases)

Loads CIFAR100 dataset.

### Aliases:

* tf.compat.v1.keras.datasets.cifar100.load\_data
* tf.compat.v2.keras.datasets.cifar100.load\_data
* tf.keras.datasets.cifar100.load\_data

tf.keras.datasets.cifar100.load\_data(label\_mode='fine')

Defined in [python/keras/datasets/cifar100.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/cifar100.py).

#### Arguments:

* **label\_mode**: one of "fine", "coarse".

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

#### Raises:

* **ValueError**: in case of invalid label\_mode.

# tf.keras.datasets.fashion\_mnist.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/fashion_mnist/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/fashion_mnist/load_data#aliases)

Loads the Fashion-MNIST dataset.

### Aliases:

* tf.compat.v1.keras.datasets.fashion\_mnist.load\_data
* tf.compat.v2.keras.datasets.fashion\_mnist.load\_data
* tf.keras.datasets.fashion\_mnist.load\_data

tf.keras.datasets.fashion\_mnist.load\_data()

Defined in [python/keras/datasets/fashion\_mnist.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/fashion_mnist.py).

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

#### License:

The copyright for Fashion-MNIST is held by Zalando SE. Fashion-MNIST is licensed under the [MIT license](https://github.com/zalandoresearch/fashion-mnist/blob/master/LICENSE).

# tf.keras.datasets.imdb.get\_word\_index

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/imdb/get_word_index#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/imdb/get_word_index#aliases)

Retrieves the dictionary mapping word indices back to words.

### Aliases:

* tf.compat.v1.keras.datasets.imdb.get\_word\_index
* tf.compat.v2.keras.datasets.imdb.get\_word\_index
* tf.keras.datasets.imdb.get\_word\_index

tf.keras.datasets.imdb.get\_word\_index(path='imdb\_word\_index.json')

Defined in [python/keras/datasets/imdb.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/imdb.py).

#### Arguments:

* **path**: where to cache the data (relative to ~/.keras/dataset).

#### Returns:

The word index dictionary.

# tf.keras.datasets.imdb.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/imdb/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/imdb/load_data#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/imdb/load_data#used_in_the_tutorials)

Loads the IMDB dataset.

### Aliases:

* tf.compat.v1.keras.datasets.imdb.load\_data
* tf.compat.v2.keras.datasets.imdb.load\_data
* tf.keras.datasets.imdb.load\_data

tf.keras.datasets.imdb.load\_data(  
    path='imdb.npz',  
    num\_words=None,  
    skip\_top=0,  
    maxlen=None,  
    seed=113,  
    start\_char=1,  
    oov\_char=2,  
    index\_from=3,  
    \*\*kwargs  
)

Defined in [python/keras/datasets/imdb.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/imdb.py).

### Used in the tutorials:

* [Explore overfitting and underfitting](https://www.tensorflow.org/beta/tutorials/keras/overfit_and_underfit)

#### Arguments:

* **path**: where to cache the data (relative to ~/.keras/dataset).
* **num\_words**: max number of words to include. Words are ranked by how often they occur (in the training set) and only the most frequent words are kept
* **skip\_top**: skip the top N most frequently occurring words (which may not be informative).
* **maxlen**: sequences longer than this will be filtered out.
* **seed**: random seed for sample shuffling.
* **start\_char**: The start of a sequence will be marked with this character. Set to 1 because 0 is usually the padding character.
* **oov\_char**: words that were cut out because of the num\_words or skip\_top limit will be replaced with this character.
* **index\_from**: index actual words with this index and higher.
* **\*\*kwargs**: Used for backwards compatibility.

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

#### Raises:

* **ValueError**: in case maxlen is so low that no input sequence could be kept.

Note that the 'out of vocabulary' character is only used for words that were present in the training set but are not included because they're not making the num\_words cut here. Words that were not seen in the training set but are in the test set have simply been skipped.

# tf.keras.datasets.mnist.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/mnist/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/mnist/load_data#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/mnist/load_data#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/mnist/load_data#used_in_the_tutorials)

Loads the MNIST dataset.

### Aliases:

* tf.compat.v1.keras.datasets.mnist.load\_data
* tf.compat.v2.keras.datasets.mnist.load\_data
* tf.keras.datasets.mnist.load\_data

tf.keras.datasets.mnist.load\_data(path='mnist.npz')

Defined in [python/keras/datasets/mnist.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/mnist.py).

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Keras custom callbacks](https://www.tensorflow.org/beta/guide/keras/custom_callback)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Convolutional Neural Networks](https://www.tensorflow.org/beta/tutorials/images/intro_to_cnns)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)

#### Arguments:

* **path**: path where to cache the dataset locally (relative to ~/.keras/datasets).

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

#### License:

Yann LeCun and Corinna Cortes hold the copyright of MNIST dataset, which is a derivative work from original NIST datasets. MNIST dataset is made available under the terms of the [Creative Commons Attribution-Share Alike 3.0 license.](https://creativecommons.org/licenses/by-sa/3.0/)

# tf.keras.datasets.reuters.get\_word\_index

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/reuters/get_word_index#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/reuters/get_word_index#aliases)

Retrieves the dictionary mapping word indices back to words.

### Aliases:

* tf.compat.v1.keras.datasets.reuters.get\_word\_index
* tf.compat.v2.keras.datasets.reuters.get\_word\_index
* tf.keras.datasets.reuters.get\_word\_index

tf.keras.datasets.reuters.get\_word\_index(path='reuters\_word\_index.json')

Defined in [python/keras/datasets/reuters.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/reuters.py).

#### Arguments:

* **path**: where to cache the data (relative to ~/.keras/dataset).

#### Returns:

The word index dictionary.

# tf.keras.datasets.reuters.load\_data

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/reuters/load_data#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/datasets/reuters/load_data#aliases)

Loads the Reuters newswire classification dataset.

### Aliases:

* tf.compat.v1.keras.datasets.reuters.load\_data
* tf.compat.v2.keras.datasets.reuters.load\_data
* tf.keras.datasets.reuters.load\_data

tf.keras.datasets.reuters.load\_data(  
    path='reuters.npz',  
    num\_words=None,  
    skip\_top=0,  
    maxlen=None,  
    test\_split=0.2,  
    seed=113,  
    start\_char=1,  
    oov\_char=2,  
    index\_from=3,  
    \*\*kwargs  
)

Defined in [python/keras/datasets/reuters.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/datasets/reuters.py).

#### Arguments:

* **path**: where to cache the data (relative to ~/.keras/dataset).
* **num\_words**: max number of words to include. Words are ranked by how often they occur (in the training set) and only the most frequent words are kept
* **skip\_top**: skip the top N most frequently occurring words (which may not be informative).
* **maxlen**: truncate sequences after this length.
* **test\_split**: Fraction of the dataset to be used as test data.
* **seed**: random seed for sample shuffling.
* **start\_char**: The start of a sequence will be marked with this character. Set to 1 because 0 is usually the padding character.
* **oov\_char**: words that were cut out because of the num\_words or skip\_top limit will be replaced with this character.
* **index\_from**: index actual words with this index and higher.
* **\*\*kwargs**: Used for backwards compatibility.

#### Returns:

Tuple of Numpy arrays: (x\_train, y\_train), (x\_test, y\_test).

Note that the 'out of vocabulary' character is only used for words that were present in the training set but are not included because they're not making the num\_words cut here. Words that were not seen in the training set but are in the test set have simply been skipped.

# tf.keras.estimator.model\_to\_estimator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/estimator/model_to_estimator#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/estimator/model_to_estimator#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/estimator/model_to_estimator#used_in_the_guide)

Constructs an Estimator instance from given keras model.

### Aliases:

* tf.compat.v2.keras.estimator.model\_to\_estimator
* tf.keras.estimator.model\_to\_estimator

tf.keras.estimator.model\_to\_estimator(  
    keras\_model=None,  
    keras\_model\_path=None,  
    custom\_objects=None,  
    model\_dir=None,  
    config=None,  
    checkpoint\_format='checkpoint'  
)

Defined in [python/keras/estimator/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/estimator/__init__.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)

For usage example, please see: [Creating estimators from Keras Models](https://tensorflow.org/guide/estimators#model_to_estimator).

#### Args:

* **keras\_model**: A compiled Keras model object. This argument is mutually exclusive with keras\_model\_path.
* **keras\_model\_path**: Path to a compiled Keras model saved on disk, in HDF5 format, which can be generated with the save() method of a Keras model. This argument is mutually exclusive with keras\_model.
* **custom\_objects**: Dictionary for custom objects.
* **model\_dir**: Directory to save Estimator model parameters, graph, summary files for TensorBoard, etc.
* **config**: RunConfig to config Estimator.
* **checkpoint\_format**: Sets the format of the checkpoint saved by the estimator when training. May be saver or checkpoint, depending on whether to save checkpoints from [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) or [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint). The default is checkpoint. Estimators use name-based tf.train.Saver checkpoints, while Keras models use object-based checkpoints from [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint). Currently, saving object-based checkpoints from model\_to\_estimator is only supported by Functional and Sequential models.

#### Returns:

An Estimator from given keras model.

#### Raises:

* **ValueError**: if neither keras\_model nor keras\_model\_path was given.
* **ValueError**: if both keras\_model and keras\_model\_path was given.
* **ValueError**: if the keras\_model\_path is a GCS URI.
* **ValueError**: if keras\_model has not been compiled.
* **ValueError**: if an invalid checkpoint\_format was given.

Module: tf.keras.experimental

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental#functions)

Public API for tf.keras.experimental namespace.

Aliases:

* Module tf.compat.v2.keras.experimental
* Module tf.keras.experimental

Defined in [python/keras/api/\_v2/keras/experimental/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/experimental/__init__.py).

Classes

[class CosineDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecay): A LearningRateSchedule that uses a cosine decay schedule.

[class CosineDecayRestarts](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecayRestarts): A LearningRateSchedule that uses a cosine decay schedule with restarts.

[class LinearCosineDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/LinearCosineDecay): A LearningRateSchedule that uses a linear cosine decay schedule.

[class NoisyLinearCosineDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/NoisyLinearCosineDecay): A LearningRateSchedule that uses a noisy linear cosine decay schedule.

[class PeepholeLSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/PeepholeLSTMCell): Equivalent to LSTMCell class but adds peephole connections.

[class SequenceFeatures](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/SequenceFeatures): A layer for sequence input.

Functions

[export\_saved\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model): Exports a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) as a Tensorflow SavedModel.

[load\_from\_saved\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/load_from_saved_model): Loads a keras Model from a SavedModel created by export\_saved\_model().

[terminate\_keras\_multiprocessing\_pools(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/terminate_keras_multiprocessing_pools): Destroy Keras' multiprocessing pools to prevent deadlocks.

# tf.keras.experimental.CosineDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecay#top_of_page)
* [Class CosineDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecay#class_cosinedecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecay#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecay#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecay#methods)

## Class CosineDecay

A LearningRateSchedule that uses a cosine decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.experimental.CosineDecay
* Class tf.compat.v2.keras.experimental.CosineDecay
* Class tf.keras.experimental.CosineDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    decay\_steps,  
    alpha=0.0,  
    name=None  
)

Applies cosine decay to the learning rate.

See [Loshchilov & Hutter, ICLR2016], SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

When training a model, it is often recommended to lower the learning rate as the training progresses. This schedule applies a cosine decay function to an optimizer step, given a provided initial learning rate. It requires a step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The schedule a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions. It is computed as:

def decayed\_learning\_rate(step):  
  step = min(step, decay\_steps)  
  cosine\_decay = 0.5 \* (1 + cos(pi \* step / decay\_steps))  
  decayed = (1 - alpha) \* cosine\_decay + alpha  
  return initial\_learning\_rate \* decayed

#### Example usage:

decay\_steps = 1000  
lr\_decayed\_fn = tf.keras.experimental.CosineDecay(  
    initial\_learning\_rate, global\_step, decay\_steps)

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. The learning rate schedule is also serializable and deserializable using [tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **alpha**: A scalar float32 or float64 Tensor or a Python number. Minimum learning rate value as a fraction of initial\_learning\_rate.
* **name**: String. Optional name of the operation. Defaults to 'CosineDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.experimental.CosineDecayRestarts

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecayRestarts#top_of_page)
* [Class CosineDecayRestarts](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecayRestarts#class_cosinedecayrestarts)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecayRestarts#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecayRestarts#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/CosineDecayRestarts#methods)

## Class CosineDecayRestarts

A LearningRateSchedule that uses a cosine decay schedule with restarts.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.experimental.CosineDecayRestarts
* Class tf.compat.v2.keras.experimental.CosineDecayRestarts
* Class tf.keras.experimental.CosineDecayRestarts

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    first\_decay\_steps,  
    t\_mul=2.0,  
    m\_mul=1.0,  
    alpha=0.0,  
    name=None  
)

Applies cosine decay with restarts to the learning rate.

See [Loshchilov & Hutter, ICLR2016], SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

When training a model, it is often recommended to lower the learning rate as the training progresses. This schedule applies a cosine decay function with restarts to an optimizer step, given a provided initial learning rate. It requires a step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The schedule a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions.

The learning rate multiplier first decays from 1 to alpha for first\_decay\_steps steps. Then, a warm restart is performed. Each new warm restart runs for t\_mul times more steps and with m\_multimes smaller initial learning rate.

#### Example usage:

first\_decay\_steps = 1000  
lr\_decayed\_fn = (  
  tf.keras.experimental.CosineDecayRestarts(  
      initial\_learning\_rate,  
      global\_step,  
      first\_decay\_steps))

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. The learning rate schedule is also serializable and deserializable using [tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **first\_decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **t\_mul**: A scalar float32 or float64 Tensor or a Python number. Used to derive the number of iterations in the i-th period
* **m\_mul**: A scalar float32 or float64 Tensor or a Python number. Used to derive the initial learning rate of the i-th period:
* **alpha**: A scalar float32 or float64 Tensor or a Python number. Minimum learning rate value as a fraction of the initial\_learning\_rate.
* **name**: String. Optional name of the operation. Defaults to 'SGDRDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.experimental.export\_saved\_model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/export_saved_model#used_in_the_tutorials)

Exports a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) as a Tensorflow SavedModel.

### Aliases:

* tf.compat.v1.keras.experimental.export\_saved\_model
* tf.compat.v2.keras.experimental.export\_saved\_model
* tf.keras.experimental.export\_saved\_model

tf.keras.experimental.export\_saved\_model(  
    model,  
    saved\_model\_path,  
    custom\_objects=None,  
    as\_text=False,  
    input\_signature=None,  
    serving\_only=False  
)

Defined in [python/keras/saving/saved\_model.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/saved_model.py).

### Used in the guide:

* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)

### Used in the tutorials:

* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)

Note that at this time, subclassed models can only be saved using serving\_only=True.

The exported SavedModel is a standalone serialization of Tensorflow objects, and is supported by TF language APIs and the Tensorflow Serving system. To load the model, use the function[tf.keras.experimental.load\_from\_saved\_model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/load_from_saved_model).

The SavedModel contains:

1. a checkpoint containing the model weights.
2. a SavedModel proto containing the Tensorflow backend graph. Separate graphs are saved for prediction (serving), train, and evaluation. If the model has not been compiled, then only the graph computing predictions will be exported.
3. the model's json config. If the model is subclassed, this will only be included if the model's get\_config() method is overwritten.

#### Example:

import tensorflow as tf  
  
# Create a tf.keras model.  
model = tf.keras.Sequential()  
model.add(tf.keras.layers.Dense(1, input\_shape=[10]))  
model.summary()  
  
# Save the tf.keras model in the SavedModel format.  
path = '/tmp/simple\_keras\_model'  
tf.keras.experimental.export\_saved\_model(model, path)  
  
# Load the saved keras model back.  
new\_model = tf.keras.experimental.load\_from\_saved\_model(path)  
new\_model.summary()

#### Args:

* **model**: A [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) to be saved. If the model is subclassed, the flag serving\_onlymust be set to True.
* **saved\_model\_path**: a string specifying the path to the SavedModel directory.
* **custom\_objects**: Optional dictionary mapping string names to custom classes or functions (e.g. custom loss functions).
* **as\_text**: bool, False by default. Whether to write the SavedModel proto in text format. Currently unavailable in serving-only mode.
* **input\_signature**: A possibly nested sequence of [tf.TensorSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorSpec) objects, used to specify the expected model inputs. See [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function) for more details.
* **serving\_only**: bool, False by default. When this is true, only the prediction graph is saved.

#### Raises:

* **NotImplementedError**: If the model is a subclassed model, and serving\_only is False.
* **ValueError**: If the input signature cannot be inferred from the model.
* **AssertionError**: If the SavedModel directory already exists and isn't empty.

# tf.keras.experimental.LinearCosineDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/LinearCosineDecay#top_of_page)
* [Class LinearCosineDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/LinearCosineDecay#class_linearcosinedecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/LinearCosineDecay#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/LinearCosineDecay#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/LinearCosineDecay#methods)

## Class LinearCosineDecay

A LearningRateSchedule that uses a linear cosine decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.experimental.LinearCosineDecay
* Class tf.compat.v2.keras.experimental.LinearCosineDecay
* Class tf.keras.experimental.LinearCosineDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    decay\_steps,  
    num\_periods=0.5,  
    alpha=0.0,  
    beta=0.001,  
    name=None  
)

Applies linear cosine decay to the learning rate.

See [Bello et al., ICML2017] Neural Optimizer Search with RL. https://arxiv.org/abs/1709.07417

For the idea of warm starts here controlled by num\_periods, see [Loshchilov & Hutter, ICLR2016] SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

Note that linear cosine decay is more aggressive than cosine decay and larger initial learning rates can typically be used.

When training a model, it is often recommended to lower the learning rate as the training progresses. This schedule applies a linear cosine decay function to an optimizer step, given a provided initial learning rate. It requires a step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The schedule a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions. It is computed as:

def decayed\_learning\_rate(step):  
  step = min(step, decay\_steps)  
  linear\_decay = (decay\_steps - step) / decay\_steps  
  cosine\_decay = 0.5 \* (  
      1 + cos(pi \* 2 \* num\_periods \* step / decay\_steps))  
  decayed = (alpha + linear\_decay) \* cosine\_decay + beta  
  return initial\_learning\_rate \* decayed

#### Example usage:

decay\_steps = 1000  
lr\_decayed\_fn = (  
  tf.keras.experimental.LinearCosineDecay(  
    initial\_learning\_rate, global\_step, decay\_steps))

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. The learning rate schedule is also serializable and deserializable using [tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **num\_periods**: Number of periods in the cosine part of the decay. See computation above.
* **alpha**: See computation above.
* **beta**: See computation above.
* **name**: String. Optional name of the operation. Defaults to 'LinearCosineDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.experimental.load\_from\_saved\_model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/load_from_saved_model#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/load_from_saved_model#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/load_from_saved_model#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/load_from_saved_model#used_in_the_tutorials)

Loads a keras Model from a SavedModel created by export\_saved\_model().

### Aliases:

* tf.compat.v1.keras.experimental.load\_from\_saved\_model
* tf.compat.v2.keras.experimental.load\_from\_saved\_model
* tf.keras.experimental.load\_from\_saved\_model

tf.keras.experimental.load\_from\_saved\_model(  
    saved\_model\_path,  
    custom\_objects=None  
)

Defined in [python/keras/saving/saved\_model.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/saved_model.py).

### Used in the guide:

* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)

### Used in the tutorials:

* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)

This function reinstantiates model state by: 1) loading model topology from json (this will eventually come from metagraph). 2) loading model weights from checkpoint.

#### Example:

import tensorflow as tf  
  
# Create a tf.keras model.  
model = tf.keras.Sequential()  
model.add(tf.keras.layers.Dense(1, input\_shape=[10]))  
model.summary()  
  
# Save the tf.keras model in the SavedModel format.  
path = '/tmp/simple\_keras\_model'  
tf.keras.experimental.export\_saved\_model(model, path)  
  
# Load the saved keras model back.  
new\_model = tf.keras.experimental.load\_from\_saved\_model(path)  
new\_model.summary()

#### Args:

* **saved\_model\_path**: a string specifying the path to an existing SavedModel.
* **custom\_objects**: Optional dictionary mapping names (strings) to custom classes or functions to be considered during deserialization.

#### Returns:

a keras.Model instance.

# tf.keras.experimental.NoisyLinearCosineDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/NoisyLinearCosineDecay#top_of_page)
* [Class NoisyLinearCosineDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/NoisyLinearCosineDecay#class_noisylinearcosinedecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/NoisyLinearCosineDecay#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/NoisyLinearCosineDecay#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/NoisyLinearCosineDecay#methods)

## Class NoisyLinearCosineDecay

A LearningRateSchedule that uses a noisy linear cosine decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.experimental.NoisyLinearCosineDecay
* Class tf.compat.v2.keras.experimental.NoisyLinearCosineDecay
* Class tf.keras.experimental.NoisyLinearCosineDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    decay\_steps,  
    initial\_variance=1.0,  
    variance\_decay=0.55,  
    num\_periods=0.5,  
    alpha=0.0,  
    beta=0.001,  
    name=None  
)

Applies noisy linear cosine decay to the learning rate.

See [Bello et al., ICML2017] Neural Optimizer Search with RL. https://arxiv.org/abs/1709.07417

For the idea of warm starts here controlled by num\_periods, see [Loshchilov & Hutter, ICLR2016] SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

Note that linear cosine decay is more aggressive than cosine decay and larger initial learning rates can typically be used.

When training a model, it is often recommended to lower the learning rate as the training progresses. This schedule applies a noisy linear cosine decay function to an optimizer step, given a provided initial learning rate. It requires a step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The schedule a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions. It is computed as:

def decayed\_learning\_rate(step):  
  step = min(step, decay\_steps)  
  linear\_decay = (decay\_steps - step) / decay\_steps)  
  cosine\_decay = 0.5 \* (  
      1 + cos(pi \* 2 \* num\_periods \* step / decay\_steps))  
  decayed = (alpha + linear\_decay + eps\_t) \* cosine\_decay + beta  
  return initial\_learning\_rate \* decayed

where eps\_t is 0-centered gaussian noise with variance initial\_variance / (1 + global\_step) \*\* variance\_decay

#### Example usage:

decay\_steps = 1000  
lr\_decayed\_fn = (  
  tf.keras.experimental.NoisyLinearCosineDecay(  
    initial\_learning\_rate, global\_step, decay\_steps))

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. The learning rate schedule is also serializable and deserializable using [tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **initial\_variance**: initial variance for the noise. See computation above.
* **variance\_decay**: decay for the noise's variance. See computation above.
* **num\_periods**: Number of periods in the cosine part of the decay. See computation above.
* **alpha**: See computation above.
* **beta**: See computation above.
* **name**: String. Optional name of the operation. Defaults to 'NoisyLinearCosineDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.experimental.PeepholeLSTMCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/PeepholeLSTMCell#top_of_page)
* [Class PeepholeLSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/PeepholeLSTMCell#class_peepholelstmcell)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/PeepholeLSTMCell#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/PeepholeLSTMCell#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/PeepholeLSTMCell#methods)

## Class PeepholeLSTMCell

Equivalent to LSTMCell class but adds peephole connections.

Inherits From: [LSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/LSTMCell)

### Aliases:

* Class tf.compat.v1.keras.experimental.PeepholeLSTMCell
* Class tf.compat.v2.keras.experimental.PeepholeLSTMCell
* Class tf.keras.experimental.PeepholeLSTMCell

Defined in [python/keras/layers/recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent.py).

Peephole connections allow the gates to utilize the previous internal state as well as the previous hidden state (which is what LSTMCell is limited to). This allows PeepholeLSTMCell to better learn precise timings over LSTMCell.

From [Gers et al.](http://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf):

"We find that LSTM augmented by 'peephole connections' from its internal cells to its multiplicative gates can learn the fine distinction between sequences of spikes spaced either 50 or 49 time steps apart without the help of any short training exemplars."

The peephole implementation is based on:

[Long short-term memory recurrent neural network architectures for large scale acoustic modeling.](https://research.google.com/pubs/archive/43905.pdf)

#### Example:

# Create 2 PeepholeLSTMCells  
peephole\_lstm\_cells = [PeepholeLSTMCell(size) for size in [128, 256]]  
# Create a layer composed sequentially of the peephole LSTM cells.  
layer = RNN(peephole\_lstm\_cells)  
input = keras.Input((timesteps, input\_dim))  
output = layer(input)

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    recurrent\_activation='hard\_sigmoid',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    unit\_forget\_bias=True,  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    implementation=1,  
    \*\*kwargs  
)

## Methods

### get\_dropout\_mask\_for\_cell

get\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the dropout mask for RNN cell's input.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

### get\_recurrent\_dropout\_mask\_for\_cell

get\_recurrent\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the recurrent dropout mask for RNN cell.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### reset\_dropout\_mask

reset\_dropout\_mask()

Reset the cached dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_recurrent\_dropout\_mask

reset\_recurrent\_dropout\_mask()

Reset the cached recurrent dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

# tf.keras.experimental.SequenceFeatures

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/SequenceFeatures#top_of_page)
* [Class SequenceFeatures](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/SequenceFeatures#class_sequencefeatures)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/SequenceFeatures#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/SequenceFeatures#__init__)

## Class SequenceFeatures

A layer for sequence input.

### Aliases:

* Class tf.compat.v1.keras.experimental.SequenceFeatures
* Class tf.compat.v2.keras.experimental.SequenceFeatures
* Class tf.keras.experimental.SequenceFeatures

Defined in [python/feature\_column/sequence\_feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/sequence_feature_column.py).

All feature\_columns must be sequence dense columns with the same sequence\_length. The output of this method can be fed into sequence networks, such as RNN.

The output of this method is a 3D Tensor of shape [batch\_size, T, D]. T is the maximum sequence length for this batch, which could differ from batch to batch.

If multiple feature\_columns are given with Di num\_elements each, their outputs are concatenated. So, the final Tensor has shape [batch\_size, T, D0 + D1 + ... + Dn].

#### Example:

rating = sequence\_numeric\_column('rating')  
watches = sequence\_categorical\_column\_with\_identity(  
    'watches', num\_buckets=1000)  
watches\_embedding = embedding\_column(watches, dimension=10)  
columns = [rating, watches\_embedding]  
  
sequence\_input\_layer = SequenceFeatures(columns)  
features = tf.io.parse\_example(...,  
                               features=make\_parse\_example\_spec(columns))  
sequence\_input, sequence\_length = sequence\_input\_layer(features)  
sequence\_length\_mask = tf.sequence\_mask(sequence\_length)  
  
rnn\_cell = tf.keras.layers.SimpleRNNCell(hidden\_size)  
rnn\_layer = tf.keras.layers.RNN(rnn\_cell)  
outputs, state = rnn\_layer(sequence\_input, mask=sequence\_length\_mask)

## \_\_init\_\_

\_\_init\_\_(  
    feature\_columns,  
    trainable=True,  
    name=None,  
    \*\*kwargs  
)

"Constructs a SequenceFeatures layer.

#### Args:

* **feature\_columns**: An iterable of dense sequence columns. Valid columns are
  + embedding\_column that wraps a sequence\_categorical\_column\_with\_\*
  + sequence\_numeric\_column.
* **trainable**: Boolean, whether the layer's variables will be updated via gradient descent during training.
* **name**: Name to give to the SequenceFeatures.
* **\*\*kwargs**: Keyword **Arguments** to construct a layer.

#### Raises:

* **ValueError**: If any of the feature\_columns is not a SequenceDenseColumn.

# tf.keras.experimental.terminate\_keras\_multiprocessing\_pools

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/terminate_keras_multiprocessing_pools#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/experimental/terminate_keras_multiprocessing_pools#aliases)

Destroy Keras' multiprocessing pools to prevent deadlocks.

### Aliases:

* tf.compat.v1.keras.experimental.terminate\_keras\_multiprocessing\_pools
* tf.compat.v2.keras.experimental.terminate\_keras\_multiprocessing\_pools
* tf.keras.experimental.terminate\_keras\_multiprocessing\_pools

tf.keras.experimental.terminate\_keras\_multiprocessing\_pools(  
    grace\_period=0.1,  
    use\_sigkill=False  
)

Defined in [python/keras/utils/data\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/data_utils.py).

In general multiprocessing.Pool can interact quite badly with other, seemingly unrelated, parts of a codebase due to Pool's reliance on fork. This method cleans up all pools which are known to belong to Keras (and thus can be safely terminated).

#### Args:

* **grace\_period**: Time (in seconds) to wait for process cleanup to propagate.
* **use\_sigkill**: Boolean of whether or not to perform a cleanup pass using SIGKILL.

#### Returns:

A list of human readable strings describing all issues encountered. It is up to the caller to decide whether to treat this as an error condition.

Module: tf.keras.initializers

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers#functions)

Keras initializer serialization / deserialization.

Aliases:

* Module tf.compat.v2.initializers
* Module tf.compat.v2.keras.initializers
* Module tf.initializers
* Module tf.keras.initializers

Defined in [python/keras/api/\_v2/keras/initializers/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/initializers/__init__.py).

Classes

[class Constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant_initializer): Initializer that generates tensors with constant values.

[class GlorotNormal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal): The Glorot normal initializer, also called Xavier normal initializer.

[class GlorotUniform](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform): The Glorot uniform initializer, also called Xavier uniform initializer.

[class Identity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity): Initializer that generates the identity matrix.

[class Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer): Initializer base class: all initializers inherit from this class.

[class Ones](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ones_initializer): Initializer that generates tensors initialized to 1.

[class Orthogonal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal): Initializer that generates an orthogonal matrix.

[class RandomNormal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random_normal_initializer): Initializer that generates tensors with a normal distribution.

[class RandomUniform](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random_uniform_initializer): Initializer that generates tensors with a uniform distribution.

[class TruncatedNormal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/TruncatedNormal): Initializer that generates a truncated normal distribution.

[class VarianceScaling](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling): Initializer capable of adapting its scale to the shape of weights tensors.

[class Zeros](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/zeros_initializer): Initializer that generates tensors initialized to 0.

[class constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant_initializer): Initializer that generates tensors with constant values.

[class glorot\_normal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal): The Glorot normal initializer, also called Xavier normal initializer.

[class glorot\_uniform](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform): The Glorot uniform initializer, also called Xavier uniform initializer.

[class identity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity): Initializer that generates the identity matrix.

[class ones](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ones_initializer): Initializer that generates tensors initialized to 1.

[class orthogonal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal): Initializer that generates an orthogonal matrix.

[class zeros](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/zeros_initializer): Initializer that generates tensors initialized to 0.

Functions

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/deserialize): Return an Initializer object from its config.

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/get)

[he\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/he_normal): He normal initializer.

[he\_uniform(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/he_uniform): He uniform variance scaling initializer.

[lecun\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/lecun_normal): LeCun normal initializer.

[lecun\_uniform(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/lecun_uniform): LeCun uniform initializer.

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/serialize)

# tf.compat.v1.keras.initializers.Constant

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Constant#top_of_page)

[Class Constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Constant#class_constant)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Constant#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Constant#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Constant#methods)

## Class Constant

Initializer that generates tensors with constant values.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Initializer)

### Aliases:

Class tf.compat.v1.constant\_initializer

Class tf.compat.v1.initializers.constant

Class tf.compat.v1.keras.initializers.Constant

Class tf.compat.v1.keras.initializers.constant

Defined in [python/ops/init\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops.py).

The resulting tensor is populated with values of type dtype, as specified by arguments valuefollowing the desired shape of the new tensor (see examples below).

The argument value can be a constant value, or a list of values of type dtype. If value is a list, then the length of the list must be less than or equal to the number of elements implied by the desired shape of the tensor. In the case where the total number of elements in value is less than the number of elements required by the tensor shape, the last element in value will be used to fill the remaining entries. If the total number of elements in value is greater than the number of elements required by the tensor shape, the initializer will raise a ValueError.

#### Args:

**value**: A Python scalar, list or tuple of values, or a N-dimensional numpy array. All elements of the initialized variable will be set to the corresponding value in the value argument.

**dtype**: Default data type, used if no dtype argument is provided when calling the initializer.

**verify\_shape**: Boolean that enables verification of the shape of value. If True, the initializer will throw an error if the shape of value is not compatible with the shape of the initialized tensor.

#### Raises:

**TypeError**: If the input value is not one of the expected types.

#### Examples:

The following example can be rewritten using a numpy.ndarray instead of the value list, even reshaped, as shown in the two commented lines below the value list initialization.

  >>> import numpy as np  
  >>> import tensorflow as tf  
  
  >>> value = [0, 1, 2, 3, 4, 5, 6, 7]  
  >>> # value = np.array(value)  
  >>> # value = value.reshape([2, 4])  
  >>> init = tf.compat.v1.constant\_initializer(value)  
  
  >>> print('fitting shape:')  
  >>> with tf.compat.v1.Session():  
  >>>   x = tf.compat.v1.get\_variable('x', shape=[2, 4], initializer=init)  
  >>>   x.initializer.run()  
  >>>   print(x.eval())  
  
  fitting shape:  
  [[ 0.  1.  2.  3.]  
   [ 4.  5.  6.  7.]]  
  
  >>> print('larger shape:')  
  >>> with tf.compat.v1.Session():  
  >>>   x = tf.compat.v1.get\_variable('x', shape=[3, 4], initializer=init)  
  >>>   x.initializer.run()  
  >>>   print(x.eval())  
  
  larger shape:  
  [[ 0.  1.  2.  3.]  
   [ 4.  5.  6.  7.]  
   [ 7.  7.  7.  7.]]  
  
  >>> print('smaller shape:')  
  >>> with tf.compat.v1.Session():  
  >>>   x = tf.compat.v1.get\_variable('x', shape=[2, 3], initializer=init)  
  
  ValueError: Too many elements provided. Needed at most 6, but received 8  
  
  >>> print('shape verification:')  
  >>> init\_verify = tf.compat.v1.constant\_initializer(value,  
  verify\_shape=True)  
  >>> with tf.compat.v1.Session():  
  >>>   x = tf.compat.v1.get\_variable('x', shape=[3, 4],  
  initializer=init\_verify)  
  
  TypeError: Expected Tensor's shape: (3, 4), got (8,).

## \_\_init\_\_

\_\_init\_\_(  
    value=0,  
    dtype=tf.dtypes.float32,  
    verify\_shape=False  
)

DEPRECATED FUNCTION ARGUMENTS (deprecated arguments)

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dtype)**. They will be removed in a future version. Instructions for updating: Call initializer instance with the dtype argument instead of passing it to the constructor**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(verify\_shape)**. They will be removed in a future version. Instructions for updating: Objects must now be the required shape or no shape can be specified

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None,  
    verify\_shape=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.compat.v1.keras.initializers.glorot\_normal

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_normal#top_of_page)

[Class glorot\_normal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_normal#class_glorot_normal)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_normal#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_normal#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_normal#methods)

## Class glorot\_normal

The Glorot normal initializer, also called Xavier normal initializer.

Inherits From: [VarianceScaling](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/VarianceScaling)

### Aliases:

Class tf.compat.v1.glorot\_normal\_initializer

Class tf.compat.v1.initializers.glorot\_normal

Class tf.compat.v1.keras.initializers.glorot\_normal

Defined in [python/ops/init\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops.py).

It draws samples from a truncated normal distribution centered on 0 with standard deviation (after truncation) given by stddev = sqrt(2 / (fan\_in + fan\_out)) where fan\_in is the number of input units in the weight tensor and fan\_out is the number of output units in the weight tensor.

#### Args:

**seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

**dtype**: Default data type, used if no dtype argument is provided when calling the initializer. Only floating point types are supported.

#### References:

[Glorot et al., 2010](http://proceedings.mlr.press/v9/glorot10a.html) ([pdf](http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf))

## \_\_init\_\_

\_\_init\_\_(  
    seed=None,  
    dtype=tf.dtypes.float32  
)

DEPRECATED FUNCTION ARGUMENTS

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dtype)**. They will be removed in a future version. Instructions for updating: Call initializer instance with the dtype argument instead of passing it to the constructor

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.compat.v1.keras.initializers.glorot\_uniform

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_uniform#top_of_page)

[Class glorot\_uniform](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_uniform#class_glorot_uniform)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_uniform#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_uniform#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/glorot_uniform#methods)

## Class glorot\_uniform

The Glorot uniform initializer, also called Xavier uniform initializer.

Inherits From: [VarianceScaling](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/VarianceScaling)

### Aliases:

Class tf.compat.v1.glorot\_uniform\_initializer

Class tf.compat.v1.initializers.glorot\_uniform

Class tf.compat.v1.keras.initializers.glorot\_uniform

Defined in [python/ops/init\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops.py).

It draws samples from a uniform distribution within [-limit, limit] where limit is sqrt(6 / (fan\_in + fan\_out)) where fan\_in is the number of input units in the weight tensor and fan\_out is the number of output units in the weight tensor.

#### Args:

**seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

**dtype**: Default data type, used if no dtype argument is provided when calling the initializer. Only floating point types are supported.

#### References:

[Glorot et al., 2010](http://proceedings.mlr.press/v9/glorot10a.html) ([pdf](http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf))

## \_\_init\_\_

\_\_init\_\_(  
    seed=None,  
    dtype=tf.dtypes.float32  
)

DEPRECATED FUNCTION ARGUMENTS

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dtype)**. They will be removed in a future version. Instructions for updating: Call initializer instance with the dtype argument instead of passing it to the constructor

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.compat.v1.keras.initializers.Ones

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Ones#top_of_page)

[Class Ones](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Ones#class_ones)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Ones#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Ones#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Ones#methods)

## Class Ones

Initializer that generates tensors initialized to 1.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Initializer)

### Aliases:

Class tf.compat.v1.initializers.ones

Class tf.compat.v1.keras.initializers.Ones

Class tf.compat.v1.keras.initializers.ones

Class tf.compat.v1.ones\_initializer

Defined in [python/ops/init\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops.py).

## \_\_init\_\_

\_\_init\_\_(dtype=tf.dtypes.float32)

DEPRECATED FUNCTION ARGUMENTS

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dtype)**. They will be removed in a future version. Instructions for updating: Call initializer instance with the dtype argument instead of passing it to the constructor

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.compat.v1.keras.initializers.RandomNormal

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomNormal#top_of_page)

[Class RandomNormal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomNormal#class_randomnormal)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomNormal#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomNormal#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomNormal#methods)

## Class RandomNormal

Initializer that generates tensors with a normal distribution.

Inherits From: [random\_normal\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/random_normal_initializer)

### Aliases:

Class tf.compat.v1.keras.initializers.RandomNormal

Class tf.compat.v1.keras.initializers.normal

Class tf.compat.v1.keras.initializers.random\_normal

Defined in [python/keras/initializers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/initializers.py).

#### Args:

**mean**: a python scalar or a scalar tensor. Mean of the random values to generate. Defaults to 0.

**stddev**: a python scalar or a scalar tensor. Standard deviation of the random values to generate. Defaults to 0.05.

**seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

**dtype**: The data type. Only floating point types are supported.

#### Returns:

RandomNormal instance.

## \_\_init\_\_

\_\_init\_\_(  
    mean=0.0,  
    stddev=0.05,  
    seed=None,  
    dtype=tf.dtypes.float32  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.compat.v1.keras.initializers.RandomUniform

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomUniform#top_of_page)

[Class RandomUniform](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomUniform#class_randomuniform)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomUniform#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomUniform#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/RandomUniform#methods)

## Class RandomUniform

Initializer that generates tensors with a uniform distribution.

Inherits From: [random\_uniform\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/random_uniform_initializer)

### Aliases:

Class tf.compat.v1.keras.initializers.RandomUniform

Class tf.compat.v1.keras.initializers.random\_uniform

Class tf.compat.v1.keras.initializers.uniform

Defined in [python/keras/initializers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/initializers.py).

#### Args:

**minval**: A python scalar or a scalar tensor. Lower bound of the range of random values to generate. Defaults to -0.05.

**maxval**: A python scalar or a scalar tensor. Upper bound of the range of random values to generate. Defaults to 0.05.

**seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

**dtype**: The data type.

#### Returns:

A RandomUniform instance.

## \_\_init\_\_

\_\_init\_\_(  
    minval=-0.05,  
    maxval=0.05,  
    seed=None,  
    dtype=tf.dtypes.float32  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.compat.v1.keras.initializers.Zeros

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Zeros#top_of_page)

[Class Zeros](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Zeros#class_zeros)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Zeros#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Zeros#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Zeros#methods)

## Class Zeros

Initializer that generates tensors initialized to 0.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/initializers/Initializer)

### Aliases:

Class tf.compat.v1.initializers.zeros

Class tf.compat.v1.keras.initializers.Zeros

Class tf.compat.v1.keras.initializers.zeros

Class tf.compat.v1.zeros\_initializer

Defined in [python/ops/init\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops.py).

## \_\_init\_\_

\_\_init\_\_(dtype=tf.dtypes.float32)

DEPRECATED FUNCTION ARGUMENTS

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dtype)**. They will be removed in a future version. Instructions for updating: Call initializer instance with the dtype argument instead of passing it to the constructor

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None,  
    partition\_info=None  
)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

**config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

tf.keras.initializers.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/deserialize#aliases)

Return an Initializer object from its config.

Aliases:

* tf.compat.v1.keras.initializers.deserialize
* tf.compat.v2.initializers.deserialize
* tf.compat.v2.keras.initializers.deserialize
* tf.initializers.deserialize
* tf.keras.initializers.deserialize

tf.keras.initializers.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/initializers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/initializers.py).

tf.keras.initializers.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/get#aliases)

Aliases:

* tf.compat.v1.keras.initializers.get
* tf.compat.v2.initializers.get
* tf.compat.v2.keras.initializers.get
* tf.initializers.get
* tf.keras.initializers.get

tf.keras.initializers.get(identifier)

Defined in [python/keras/initializers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/initializers.py).

# tf.keras.initializers.GlorotNormal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#top_of_page)
* [Class GlorotNormal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#class_glorotnormal)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#__call__)
  + [from\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#from_config)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotNormal#get_config)

## Class GlorotNormal

The Glorot normal initializer, also called Xavier normal initializer.

Inherits From: [VarianceScaling](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling)

### Aliases:

* Class tf.compat.v2.initializers.GlorotNormal
* Class tf.compat.v2.initializers.glorot\_normal
* Class tf.compat.v2.keras.initializers.GlorotNormal
* Class tf.compat.v2.keras.initializers.glorot\_normal
* Class tf.initializers.GlorotNormal
* Class tf.initializers.glorot\_normal
* Class tf.keras.initializers.GlorotNormal
* Class tf.keras.initializers.glorot\_normal

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

It draws samples from a truncated normal distribution centered on 0 with stddev = sqrt(2 / (fan\_in + fan\_out)) where fan\_in is the number of input units in the weight tensor and fan\_outis the number of output units in the weight tensor.

#### Args:

* **seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### References:

[Glorot et al., 2010](http://proceedings.mlr.press/v9/glorot10a.html) ([pdf](http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf))

## \_\_init\_\_

\_\_init\_\_(seed=None)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=tf.dtypes.float32  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. Only floating point types are supported.

#### Raises:

* **ValueError**: If the dtype is not floating point

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.keras.initializers.GlorotUniform

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform#top_of_page)
* [Class GlorotUniform](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform#class_glorotuniform)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/GlorotUniform#methods)

## Class GlorotUniform

The Glorot uniform initializer, also called Xavier uniform initializer.

Inherits From: [VarianceScaling](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling)

### Aliases:

* Class tf.compat.v2.initializers.GlorotUniform
* Class tf.compat.v2.initializers.glorot\_uniform
* Class tf.compat.v2.keras.initializers.GlorotUniform
* Class tf.compat.v2.keras.initializers.glorot\_uniform
* Class tf.initializers.GlorotUniform
* Class tf.initializers.glorot\_uniform
* Class tf.keras.initializers.GlorotUniform
* Class tf.keras.initializers.glorot\_uniform

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

It draws samples from a uniform distribution within [-limit, limit] where limit is sqrt(6 / (fan\_in + fan\_out)) where fan\_in is the number of input units in the weight tensor and fan\_out is the number of output units in the weight tensor.

#### Args:

* **seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### References:

[Glorot et al., 2010](http://proceedings.mlr.press/v9/glorot10a.html) ([pdf](http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf))

## \_\_init\_\_

\_\_init\_\_(seed=None)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=tf.dtypes.float32  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. Only floating point types are supported.

#### Raises:

* **ValueError**: If the dtype is not floating point

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.keras.initializers.he\_normal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/he_normal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/he_normal#aliases)

He normal initializer.

### Aliases:

* tf.compat.v2.initializers.he\_normal
* tf.compat.v2.keras.initializers.he\_normal
* tf.initializers.he\_normal
* tf.keras.initializers.he\_normal

tf.keras.initializers.he\_normal(seed=None)

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

It draws samples from a truncated normal distribution centered on 0 with stddev = sqrt(2 / fan\_in) where fan\_in is the number of input units in the weight tensor.

#### Arguments:

* **seed**: A Python integer. Used to seed the random generator.

#### Returns:

An initializer.

#### References:

[He et al., 2015](https://www.cv-foundation.org/openaccess/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html) # pylint: disable=line-too-long ([pdf](https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/He_Delving_Deep_into_ICCV_2015_paper.pdf))

# tf.keras.initializers.he\_uniform

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/he_uniform#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/he_uniform#aliases)

He uniform variance scaling initializer.

### Aliases:

* tf.compat.v2.initializers.he\_uniform
* tf.compat.v2.keras.initializers.he\_uniform
* tf.initializers.he\_uniform
* tf.keras.initializers.he\_uniform

tf.keras.initializers.he\_uniform(seed=None)

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

It draws samples from a uniform distribution within [-limit, limit] where limit is sqrt(6 / fan\_in)where fan\_in is the number of input units in the weight tensor.

#### Arguments:

* **seed**: A Python integer. Used to seed the random generator.

#### Returns:

An initializer.

#### References:

[He et al., 2015](https://www.cv-foundation.org/openaccess/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html) # pylint: disable=line-too-long ([pdf](https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/He_Delving_Deep_into_ICCV_2015_paper.pdf))

# tf.keras.initializers.Identity

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity#top_of_page)
* [Class Identity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity#class_identity)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Identity#methods)

## Class Identity

Initializer that generates the identity matrix.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer)

### Aliases:

* Class tf.compat.v2.initializers.Identity
* Class tf.compat.v2.initializers.identity
* Class tf.compat.v2.keras.initializers.Identity
* Class tf.compat.v2.keras.initializers.identity
* Class tf.initializers.Identity
* Class tf.initializers.identity
* Class tf.keras.initializers.Identity
* Class tf.keras.initializers.identity

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

Only use for 2D matrices.

#### Args:

* **gain**: Multiplicative factor to apply to the identity matrix.

## \_\_init\_\_

\_\_init\_\_(gain=1.0)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=tf.dtypes.float32  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. Only floating point types are supported.

#### Raises:

* **ValueError**: If the dtype is not floating point

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.keras.initializers.Initializer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#top_of_page)
* [Class Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#class_initializer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#__call__)
  + [from\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#from_config)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer#get_config)

## Class Initializer

Initializer base class: all initializers inherit from this class.

### Aliases:

* Class tf.compat.v2.initializers.Initializer
* Class tf.compat.v2.keras.initializers.Initializer
* Class tf.initializers.Initializer
* Class tf.keras.initializers.Initializer

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=None  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. If not provided will return tensor of [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32).

### from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

Returns the configuration of the initializer as a JSON-serializable dict.

#### Returns:

A JSON-serializable Python dict.

# tf.keras.initializers.lecun\_normal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/lecun_normal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/lecun_normal#aliases)

LeCun normal initializer.

### Aliases:

* tf.compat.v2.initializers.lecun\_normal
* tf.compat.v2.keras.initializers.lecun\_normal
* tf.initializers.lecun\_normal
* tf.keras.initializers.lecun\_normal

tf.keras.initializers.lecun\_normal(seed=None)

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

It draws samples from a truncated normal distribution centered on 0 with stddev = sqrt(1 / fan\_in) where fan\_in is the number of input units in the weight tensor.

#### Arguments:

* **seed**: A Python integer. Used to seed the random generator.

#### Returns:

An initializer.

#### References:

* Self-Normalizing Neural Networks, [Klambauer et al., 2017](https://papers.nips.cc/paper/6698-self-normalizing-neural-networks) ([pdf](https://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf) )
* Efficient Backprop, [Lecun et al., 1998](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)

# tf.keras.initializers.lecun\_uniform

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/lecun_uniform#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/lecun_uniform#aliases)

LeCun uniform initializer.

### Aliases:

* tf.compat.v2.initializers.lecun\_uniform
* tf.compat.v2.keras.initializers.lecun\_uniform
* tf.initializers.lecun\_uniform
* tf.keras.initializers.lecun\_uniform

tf.keras.initializers.lecun\_uniform(seed=None)

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

It draws samples from a uniform distribution within [-limit, limit] where limit is sqrt(3 / fan\_in)where fan\_in is the number of input units in the weight tensor.

#### Arguments:

* **seed**: A Python integer. Used to seed the random generator.

#### Returns:

An initializer.

#### References:

* Self-Normalizing Neural Networks, [Klambauer et al., 2017](https://papers.nips.cc/paper/6698-self-normalizing-neural-networks) # pylint: disable=line-too-long ([pdf](https://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf))
* Efficient Backprop, [Lecun et al., 1998](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)

# tf.keras.initializers.Orthogonal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal#top_of_page)
* [Class Orthogonal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal#class_orthogonal)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Orthogonal#methods)

## Class Orthogonal

Initializer that generates an orthogonal matrix.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer)

### Aliases:

* Class tf.compat.v2.initializers.Orthogonal
* Class tf.compat.v2.initializers.orthogonal
* Class tf.compat.v2.keras.initializers.Orthogonal
* Class tf.compat.v2.keras.initializers.orthogonal
* Class tf.initializers.Orthogonal
* Class tf.initializers.orthogonal
* Class tf.keras.initializers.Orthogonal
* Class tf.keras.initializers.orthogonal

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

If the shape of the tensor to initialize is two-dimensional, it is initialized with an orthogonal matrix obtained from the QR decomposition of a matrix of random numbers drawn from a normal distribution. If the matrix has fewer rows than columns then the output will have orthogonal rows. Otherwise, the output will have orthogonal columns.

If the shape of the tensor to initialize is more than two-dimensional, a matrix of shape (shape[0] \* ... \* shape[n - 2], shape[n - 1]) is initialized, where n is the length of the shape vector. The matrix is subsequently reshaped to give a tensor of the desired shape.

#### Args:

* **gain**: multiplicative factor to apply to the orthogonal matrix
* **seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### References:

[Saxe et al., 2014](https://openreview.net/forum?id=_wzZwKpTDF_9C) ([pdf](https://arxiv.org/pdf/1312.6120.pdf))

## \_\_init\_\_

\_\_init\_\_(  
    gain=1.0,  
    seed=None  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=tf.dtypes.float32  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. Only floating point types are supported.

#### Raises:

* **ValueError**: If the dtype is not floating point or the input shape is not valid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

tf.keras.initializers.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/serialize#aliases)

Aliases:

* tf.compat.v1.keras.initializers.serialize
* tf.compat.v2.initializers.serialize
* tf.compat.v2.keras.initializers.serialize
* tf.initializers.serialize
* tf.keras.initializers.serialize

tf.keras.initializers.serialize(initializer)

Defined in [python/keras/initializers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/initializers.py).

# tf.keras.initializers.TruncatedNormal

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/TruncatedNormal#top_of_page)
* [Class TruncatedNormal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/TruncatedNormal#class_truncatednormal)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/TruncatedNormal#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/TruncatedNormal#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/TruncatedNormal#methods)

## Class TruncatedNormal

Initializer that generates a truncated normal distribution.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer)

### Aliases:

* Class tf.compat.v2.initializers.TruncatedNormal
* Class tf.compat.v2.keras.initializers.TruncatedNormal
* Class tf.initializers.TruncatedNormal
* Class tf.keras.initializers.TruncatedNormal

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

These values are similar to values from a random\_normal\_initializer except that values more than two standard deviations from the mean are discarded and re-drawn. This is the recommended initializer for neural network weights and filters.

#### Args:

* **mean**: a python scalar or a scalar tensor. Mean of the random values to generate.
* **stddev**: a python scalar or a scalar tensor. Standard deviation of the random values to generate.
* **seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

## \_\_init\_\_

\_\_init\_\_(  
    mean=0.0,  
    stddev=0.05,  
    seed=None  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=tf.dtypes.float32  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. Only floating point types are supported.

#### Raises:

* **ValueError**: If the dtype is not floating point

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

# tf.keras.initializers.VarianceScaling

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling#top_of_page)
* [Class VarianceScaling](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling#class_variancescaling)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/VarianceScaling#methods)

## Class VarianceScaling

Initializer capable of adapting its scale to the shape of weights tensors.

Inherits From: [Initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/initializers/Initializer)

### Aliases:

* Class tf.compat.v2.initializers.VarianceScaling
* Class tf.compat.v2.keras.initializers.VarianceScaling
* Class tf.initializers.VarianceScaling
* Class tf.keras.initializers.VarianceScaling

Defined in [python/ops/init\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/init_ops_v2.py).

With distribution="truncated\_normal" or "untruncated\_normal", samples are drawn from a truncated/untruncated normal distribution with a mean of zero and a standard deviation (after truncation, if used) stddev = sqrt(scale / n) where n is: - number of input units in the weight tensor, if mode = "fan\_in" - number of output units, if mode = "fan\_out" - average of the numbers of input and output units, if mode = "fan\_avg"

With distribution="uniform", samples are drawn from a uniform distribution within [-limit, limit], with limit = sqrt(3 \* scale / n).

#### Args:

* **scale**: Scaling factor (positive float).
* **mode**: One of "fan\_in", "fan\_out", "fan\_avg".
* **distribution**: Random distribution to use. One of "truncated\_normal", "untruncated\_normal" and "uniform".
* **seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Raises:

* **ValueError**: In case of an invalid value for the "scale", mode" or "distribution" **Arguments**.

## \_\_init\_\_

\_\_init\_\_(  
    scale=1.0,  
    mode='fan\_in',  
    distribution='truncated\_normal',  
    seed=None  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    shape,  
    dtype=tf.dtypes.float32  
)

Returns a tensor object initialized as specified by the initializer.

#### Args:

* **shape**: Shape of the tensor.
* **dtype**: Optional dtype of the tensor. Only floating point types are supported.

#### Raises:

* **ValueError**: If the dtype is not floating point

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates an initializer from a configuration dictionary.

#### Example:

initializer = RandomUniform(-1, 1)  
config = initializer.get\_config()  
initializer = RandomUniform.from\_config(config)

#### Args:

* **config**: A Python dictionary. It will typically be the output of get\_config.

#### Returns:

An Initializer instance.

### get\_config

get\_config()

Module: tf.compat.v1.layers

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers#top_of_page)

[Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers#modules)

[Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers#classes)

[Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers#functions)

Public API for tf.layers namespace.

Modules

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/experimental) module: Public API for tf.layers.experimental namespace.

Classes

[class AveragePooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/AveragePooling1D): Average Pooling layer for 1D inputs.

[class AveragePooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/AveragePooling2D): Average pooling layer for 2D inputs (e.g. images).

[class AveragePooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/AveragePooling3D): Average pooling layer for 3D inputs (e.g. volumes).

[class BatchNormalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/BatchNormalization): Batch Normalization layer from http://arxiv.org/abs/1502.03167.

[class Conv1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Conv1D): 1D convolution layer (e.g. temporal convolution).

[class Conv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Conv2D): 2D convolution layer (e.g. spatial convolution over images).

[class Conv2DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Conv2DTranspose): Transposed 2D convolution layer (sometimes called 2D Deconvolution).

[class Conv3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Conv3D): 3D convolution layer (e.g. spatial convolution over volumes).

[class Conv3DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Conv3DTranspose): Transposed 3D convolution layer (sometimes called 3D Deconvolution).

[class Dense](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Dense): Densely-connected layer class.

[class Dropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Dropout): Applies Dropout to the input.

[class Flatten](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Flatten): Flattens an input tensor while preserving the batch axis (axis 0).

[class InputSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec): Specifies the ndim, dtype and shape of every input to a layer.

[class Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/Layer): Base layer class.

[class MaxPooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/MaxPooling1D): Max Pooling layer for 1D inputs.

[class MaxPooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/MaxPooling2D): Max pooling layer for 2D inputs (e.g. images).

[class MaxPooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/MaxPooling3D): Max pooling layer for 3D inputs (e.g. volumes).

[class SeparableConv1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/SeparableConv1D): Depthwise separable 1D convolution.

[class SeparableConv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/SeparableConv2D): Depthwise separable 2D convolution.

Functions

[average\_pooling1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/average_pooling1d): Average Pooling layer for 1D inputs. (deprecated)

[average\_pooling2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/average_pooling2d): Average pooling layer for 2D inputs (e.g. images). (deprecated)

[average\_pooling3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/average_pooling3d): Average pooling layer for 3D inputs (e.g. volumes). (deprecated)

[batch\_normalization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/batch_normalization): Functional interface for the batch normalization layer. (deprecated)

[conv1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/conv1d): Functional interface for 1D convolution layer (e.g. temporal convolution). (deprecated)

[conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/conv2d): Functional interface for the 2D convolution layer. (deprecated)

[conv2d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/conv2d_transpose): Functional interface for transposed 2D convolution layer. (deprecated)

[conv3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/conv3d): Functional interface for the 3D convolution layer. (deprecated)

[conv3d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/conv3d_transpose): Functional interface for transposed 3D convolution layer. (deprecated)

[dense(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/dense): Functional interface for the densely-connected layer. (deprecated)

[dropout(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/dropout): Applies Dropout to the input. (deprecated)

[flatten(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/flatten): Flattens an input tensor while preserving the batch axis (axis 0). (deprecated)

[max\_pooling1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/max_pooling1d): Max Pooling layer for 1D inputs. (deprecated)

[max\_pooling2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/max_pooling2d): Max pooling layer for 2D inputs (e.g. images). (deprecated)

[max\_pooling3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/max_pooling3d): Max pooling layer for 3D inputs (e.g. (deprecated)

[separable\_conv1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/separable_conv1d): Functional interface for the depthwise separable 1D convolution layer. (deprecated)

[separable\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/layers/separable_conv2d): Functional interface for the depthwise separable 2D convolution layer. (deprecated)

Module: tf.keras.layers

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers#functions)

Keras layers API.

Aliases:

* Module tf.compat.v2.keras.layers
* Module tf.keras.layers

Defined in [python/keras/api/\_v2/keras/layers/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/layers/__init__.py).

Classes

[class AbstractRNNCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AbstractRNNCell): Abstract object representing an RNN cell.

[class Activation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Activation): Applies an activation function to an output.

[class ActivityRegularization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ActivityRegularization): Layer that applies an update to the cost function based input activity.

[class Add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Add): Layer that adds a list of inputs.

[class AdditiveAttention](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AdditiveAttention): Additive attention layer, a.k.a. Bahdanau-style attention.

[class AlphaDropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AlphaDropout): Applies Alpha Dropout to the input.

[class Attention](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Attention): Dot-product attention layer, a.k.a. Luong-style attention.

[class Average](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Average): Layer that averages a list of inputs.

[class AveragePooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling1D): Average pooling for temporal data.

[class AveragePooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling2D): Average pooling operation for spatial data.

[class AveragePooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling3D): Average pooling operation for 3D data (spatial or spatio-temporal).

[class AvgPool1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling1D): Average pooling for temporal data.

[class AvgPool2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling2D): Average pooling operation for spatial data.

[class AvgPool3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling3D): Average pooling operation for 3D data (spatial or spatio-temporal).

[class BatchNormalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization): Base class of Batch normalization layer (Ioffe and Szegedy, 2014).

[class Bidirectional](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Bidirectional): Bidirectional wrapper for RNNs.

[class Concatenate](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Concatenate): Layer that concatenates a list of inputs.

[class Conv1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D): 1D convolution layer (e.g. temporal convolution).

[class Conv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D): 2D convolution layer (e.g. spatial convolution over images).

[class Conv2DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose): Transposed convolution layer (sometimes called Deconvolution).

[class Conv3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D): 3D convolution layer (e.g. spatial convolution over volumes).

[class Conv3DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3DTranspose): Transposed convolution layer (sometimes called Deconvolution).

[class ConvLSTM2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ConvLSTM2D): Convolutional LSTM.

[class Convolution1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D): 1D convolution layer (e.g. temporal convolution).

[class Convolution2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D): 2D convolution layer (e.g. spatial convolution over images).

[class Convolution2DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose): Transposed convolution layer (sometimes called Deconvolution).

[class Convolution3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D): 3D convolution layer (e.g. spatial convolution over volumes).

[class Convolution3DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3DTranspose): Transposed convolution layer (sometimes called Deconvolution).

[class Cropping1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping1D): Cropping layer for 1D input (e.g. temporal sequence).

[class Cropping2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping2D): Cropping layer for 2D input (e.g. picture).

[class Cropping3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping3D): Cropping layer for 3D data (e.g. spatial or spatio-temporal).

[class Dense](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense): Just your regular densely-connected NN layer.

[class DenseFeatures](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DenseFeatures): A layer that produces a dense Tensor based on given feature\_columns.

[class DepthwiseConv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DepthwiseConv2D): Depthwise separable 2D convolution.

[class Dot](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dot): Layer that computes a dot product between samples in two tensors.

[class Dropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout): Applies Dropout to the input.

[class ELU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ELU): Exponential Linear Unit.

[class Embedding](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding): Turns positive integers (indexes) into dense vectors of fixed size.

[class Flatten](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten): Flattens the input. Does not affect the batch size.

[class GRU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRU): Gated Recurrent Unit - Cho et al. 2014.

[class GRUCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRUCell): Cell class for the GRU layer.

[class GaussianDropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianDropout): Apply multiplicative 1-centered Gaussian noise.

[class GaussianNoise](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianNoise): Apply additive zero-centered Gaussian noise.

[class GlobalAveragePooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D): Global average pooling operation for temporal data.

[class GlobalAveragePooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D): Global average pooling operation for spatial data.

[class GlobalAveragePooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling3D): Global Average pooling operation for 3D data.

[class GlobalAvgPool1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D): Global average pooling operation for temporal data.

[class GlobalAvgPool2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D): Global average pooling operation for spatial data.

[class GlobalAvgPool3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling3D): Global Average pooling operation for 3D data.

[class GlobalMaxPool1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D): Global max pooling operation for temporal data.

[class GlobalMaxPool2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D): Global max pooling operation for spatial data.

[class GlobalMaxPool3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool3D): Global Max pooling operation for 3D data.

[class GlobalMaxPooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D): Global max pooling operation for temporal data.

[class GlobalMaxPooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D): Global max pooling operation for spatial data.

[class GlobalMaxPooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool3D): Global Max pooling operation for 3D data.

[class InputLayer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputLayer): Layer to be used as an entry point into a Network (a graph of layers).

[class InputSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec): Specifies the ndim, dtype and shape of every input to a layer.

[class LSTM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM): Long Short-Term Memory layer - Hochreiter 1997.

[class LSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTMCell): Cell class for the LSTM layer.

[class Lambda](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Lambda): Wraps arbitrary expressions as a Layer object.

[class Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer): Base layer class.

[class LayerNormalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LayerNormalization): Layer normalization layer (Ba et al., 2016).

[class LeakyReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LeakyReLU): Leaky version of a Rectified Linear Unit.

[class LocallyConnected1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected1D): Locally-connected layer for 1D inputs.

[class LocallyConnected2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected2D): Locally-connected layer for 2D inputs.

[class Masking](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Masking): Masks a sequence by using a mask value to skip timesteps.

[class MaxPool1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool1D): Max pooling operation for temporal data.

[class MaxPool2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D): Max pooling operation for spatial data.

[class MaxPool3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool3D): Max pooling operation for 3D data (spatial or spatio-temporal).

[class MaxPooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool1D): Max pooling operation for temporal data.

[class MaxPooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D): Max pooling operation for spatial data.

[class MaxPooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool3D): Max pooling operation for 3D data (spatial or spatio-temporal).

[class Maximum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Maximum): Layer that computes the maximum (element-wise) a list of inputs.

[class Minimum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Minimum): Layer that computes the minimum (element-wise) a list of inputs.

[class Multiply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Multiply): Layer that multiplies (element-wise) a list of inputs.

[class PReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/PReLU): Parametric Rectified Linear Unit.

[class Permute](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Permute): Permutes the dimensions of the input according to a given pattern.

[class RNN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN): Base class for recurrent layers.

[class ReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ReLU): Rectified Linear Unit activation function.

[class RepeatVector](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RepeatVector): Repeats the input n times.

[class Reshape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape): Reshapes an output to a certain shape.

[class SeparableConv1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv1D): Depthwise separable 1D convolution.

[class SeparableConv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv2D): Depthwise separable 2D convolution.

[class SeparableConvolution1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv1D): Depthwise separable 1D convolution.

[class SeparableConvolution2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv2D): Depthwise separable 2D convolution.

[class SimpleRNN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNN): Fully-connected RNN where the output is to be fed back to input.

[class SimpleRNNCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNNCell): Cell class for SimpleRNN.

[class Softmax](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Softmax): Softmax activation function.

[class SpatialDropout1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout1D): Spatial 1D version of Dropout.

[class SpatialDropout2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout2D): Spatial 2D version of Dropout.

[class SpatialDropout3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout3D): Spatial 3D version of Dropout.

[class StackedRNNCells](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/StackedRNNCells): Wrapper allowing a stack of RNN cells to behave as a single cell.

[class Subtract](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Subtract): Layer that subtracts two inputs.

[class ThresholdedReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ThresholdedReLU): Thresholded Rectified Linear Unit.

[class TimeDistributed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/TimeDistributed): This wrapper allows to apply a layer to every temporal slice of an input.

[class UpSampling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling1D): Upsampling layer for 1D inputs.

[class UpSampling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling2D): Upsampling layer for 2D inputs.

[class UpSampling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling3D): Upsampling layer for 3D inputs.

[class Wrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper): Abstract wrapper base class.

[class ZeroPadding1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding1D): Zero-padding layer for 1D input (e.g. temporal sequence).

[class ZeroPadding2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding2D): Zero-padding layer for 2D input (e.g. picture).

[class ZeroPadding3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding3D): Zero-padding layer for 3D data (spatial or spatio-temporal).

Functions

[Input(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Input): Input() is used to instantiate a Keras tensor.

[add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/add): Functional interface to the Add layer.

[average(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/average): Functional interface to the Average layer.

[concatenate(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/concatenate): Functional interface to the Concatenate layer.

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/deserialize): Instantiates a layer from a config dictionary.

[dot(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/dot): Functional interface to the Dot layer.

[maximum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/maximum): Functional interface to the Maximum layer.

[minimum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/minimum): Functional interface to the Minimum layer.

[multiply(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/multiply): Functional interface to the Multiply layer.

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/serialize)

[subtract(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/subtract): Functional interface to the Subtract layer.

# tf.compat.v1.keras.layers.CuDNNGRU

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#top_of_page)

[Class CuDNNGRU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#class_cudnngru)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#__init__)

[Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#properties)

[cell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#cell)

## Class CuDNNGRU

Fast GRU implementation backed by cuDNN.

Defined in [python/keras/layers/cudnn\_recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/cudnn_recurrent.py).

More information about cuDNN can be found on the [NVIDIA developer website](https://developer.nvidia.com/cudnn). Can only be run on GPU.

#### Arguments:

**units**: Positive integer, dimensionality of the output space.

**kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.

**recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.

**bias\_initializer**: Initializer for the bias vector.

**kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.

**recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.

**bias\_regularizer**: Regularizer function applied to the bias vector.

**activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").

**kernel\_constraint**: Constraint function applied to the kernel weights matrix.

**recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.

**bias\_constraint**: Constraint function applied to the bias vector.

**return\_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence.

**return\_state**: Boolean. Whether to return the last state in addition to the output.

**go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.

**stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    \*\*kwargs  
)

## Properties

### cell

### states

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states

reset\_states(states=None)

# tf.compat.v1.keras.layers.CuDNNLSTM

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#top_of_page)

[Class CuDNNLSTM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#class_cudnnlstm)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#__init__)

[Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#properties)

[cell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#cell)

## Class CuDNNLSTM

Fast LSTM implementation backed by cuDNN.

Defined in [python/keras/layers/cudnn\_recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/cudnn_recurrent.py).

More information about cuDNN can be found on the [NVIDIA developer website](https://developer.nvidia.com/cudnn). Can only be run on GPU.

#### Arguments:

**units**: Positive integer, dimensionality of the output space.

**kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.

**unit\_forget\_bias**: Boolean. If True, add 1 to the bias of the forget gate at initialization. Setting it to true will also force bias\_initializer="zeros". This is recommended in [Jozefowicz et al.](http://www.jmlr.org/proceedings/papers/v37/jozefowicz15.pdf)

**recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.

**bias\_initializer**: Initializer for the bias vector.

**kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.

**recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.

**bias\_regularizer**: Regularizer function applied to the bias vector.

**activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").

**kernel\_constraint**: Constraint function applied to the kernel weights matrix.

**recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.

**bias\_constraint**: Constraint function applied to the bias vector.

**return\_sequences**: Boolean. Whether to return the last output. in the output sequence, or the full sequence.

**return\_state**: Boolean. Whether to return the last state in addition to the output.

**go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.

**stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    unit\_forget\_bias=True,  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    \*\*kwargs  
)

## Properties

### cell

### states

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states

reset\_states(states=None)

# tf.keras.layers.AbstractRNNCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AbstractRNNCell#top_of_page)
* [Class AbstractRNNCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AbstractRNNCell#class_abstractrnncell)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AbstractRNNCell#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AbstractRNNCell#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AbstractRNNCell#properties)

## Class AbstractRNNCell

Abstract object representing an RNN cell.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.AbstractRNNCell
* Class tf.compat.v2.keras.layers.AbstractRNNCell
* Class tf.keras.layers.AbstractRNNCell

Defined in [python/keras/layers/recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent.py).

This is the base class for implementing RNN cells with custom behavior.

Every RNNCell must have the properties below and implement call with the signature (output, next\_state) = call(input, state).

#### Examples:

  class MinimalRNNCell(AbstractRNNCell):  
  
    def \_\_init\_\_(self, units, \*\*kwargs):  
      self.units = units  
      super(MinimalRNNCell, self).\_\_init\_\_(\*\*kwargs)  
  
    @property  
    def state\_size(self):  
      return self.units  
  
    def build(self, input\_shape):  
      self.kernel = self.add\_weight(shape=(input\_shape[-1], self.units),  
                                    initializer='uniform',  
                                    name='kernel')  
      self.recurrent\_kernel = self.add\_weight(  
          shape=(self.units, self.units),  
          initializer='uniform',  
          name='recurrent\_kernel')  
      self.built = True  
  
    def call(self, inputs, states):  
      prev\_output = states[0]  
      h = K.dot(inputs, self.kernel)  
      output = h + K.dot(prev\_output, self.recurrent\_kernel)  
      return output, output

This definition of cell differs from the definition used in the literature. In the literature, 'cell' refers to an object with a single scalar output. This definition refers to a horizontal array of such units.

An RNN cell, in the most abstract setting, is anything that has a state and performs some operation that takes a matrix of inputs. This operation results in an output matrix with self.output\_sizecolumns. If self.state\_size is an integer, this operation also results in a new state matrix with self.state\_size columns. If self.state\_size is a (possibly nested tuple of) TensorShape object(s), then it should return a matching structure of Tensors having shape [batch\_size].concatenate(s) for each s in self.batch\_size.

## \_\_init\_\_

\_\_init\_\_(  
    trainable=True,  
    name=None,  
    dtype=None,  
    dynamic=False,  
    \*\*kwargs  
)

## Properties

### output\_size

Integer or TensorShape: size of outputs produced by this cell.

### state\_size

size(s) of state(s) used by this cell.

It can be represented by an Integer, a TensorShape or a tuple of Integers or TensorShapes.

## Methods

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

# tf.keras.layers.Activation

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Activation#top_of_page)
* [Class Activation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Activation#class_activation)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Activation#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Activation#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Activation#__init__)

## Class Activation

Applies an activation function to an output.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Activation
* Class tf.compat.v2.keras.layers.Activation
* Class tf.keras.layers.Activation

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)

#### Arguments:

* **activation**: Activation function, such as [tf.nn.relu](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/relu), or string name of built-in activation function, such as "relu".

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

## \_\_init\_\_

\_\_init\_\_(  
    activation,  
    \*\*kwargs  
)

# tf.keras.layers.ActivityRegularization

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ActivityRegularization#top_of_page)
* [Class ActivityRegularization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ActivityRegularization#class_activityregularization)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ActivityRegularization#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ActivityRegularization#__init__)

## Class ActivityRegularization

Layer that applies an update to the cost function based input activity.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ActivityRegularization
* Class tf.compat.v2.keras.layers.ActivityRegularization
* Class tf.keras.layers.ActivityRegularization

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

#### Arguments:

* **l1**: L1 regularization factor (positive float).
* **l2**: L2 regularization factor (positive float).

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

## \_\_init\_\_

\_\_init\_\_(  
    l1=0.0,  
    l2=0.0,  
    \*\*kwargs  
)

# tf.keras.layers.Add

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Add#top_of_page)
* [Class Add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Add#class_add)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Add#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Add#__init__)

## Class Add

Layer that adds a list of inputs.

### Aliases:

* Class tf.compat.v1.keras.layers.Add
* Class tf.compat.v2.keras.layers.Add
* Class tf.keras.layers.Add

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

It takes as input a list of tensors, all of the same shape, and returns a single tensor (also of the same shape).

#### Examples:

    import keras  
  
    input1 = keras.layers.Input(shape=(16,))  
    x1 = keras.layers.Dense(8, activation='relu')(input1)  
    input2 = keras.layers.Input(shape=(32,))  
    x2 = keras.layers.Dense(8, activation='relu')(input2)  
    # equivalent to `added = keras.layers.add([x1, x2])`  
    added = keras.layers.Add()([x1, x2])  
    out = keras.layers.Dense(4)(added)  
    model = keras.models.Model(inputs=[input1, input2], outputs=out)

## \_\_init\_\_

\_\_init\_\_(\*\*kwargs)

# tf.keras.layers.add

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/add#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/add#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/add#used_in_the_guide)

Functional interface to the Add layer.

### Aliases:

* tf.compat.v1.keras.layers.add
* tf.compat.v2.keras.layers.add
* tf.keras.layers.add

tf.keras.layers.add(  
    inputs,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the sum of the inputs.

#### Examples:

    import keras  
  
    input1 = keras.layers.Input(shape=(16,))  
    x1 = keras.layers.Dense(8, activation='relu')(input1)  
    input2 = keras.layers.Input(shape=(32,))  
    x2 = keras.layers.Dense(8, activation='relu')(input2)  
    added = keras.layers.add([x1, x2])  
  
    out = keras.layers.Dense(4)(added)  
    model = keras.models.Model(inputs=[input1, input2], outputs=out)

# tf.keras.layers.AdditiveAttention

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AdditiveAttention#top_of_page)
* [Class AdditiveAttention](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AdditiveAttention#class_additiveattention)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AdditiveAttention#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AdditiveAttention#__init__)

## Class AdditiveAttention

Additive attention layer, a.k.a. Bahdanau-style attention.

### Aliases:

* Class tf.compat.v1.keras.layers.AdditiveAttention
* Class tf.compat.v2.keras.layers.AdditiveAttention
* Class tf.keras.layers.AdditiveAttention

Defined in [python/keras/layers/dense\_attention.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/dense_attention.py).

Inputs are query tensor of shape [batch\_size, Tq, dim], value tensor of shape [batch\_size, Tv, dim] and key tensor of shape [batch\_size, Tv, dim]. The calculation follows the steps:

1. Reshape query and value into shapes [batch\_size, Tq, 1, dim] and [batch\_size, 1, Tv, dim] respectively.
2. Calculate scores with shape [batch\_size, Tq, Tv] as a non-linear sum: scores = tf.reduce\_sum(tf.tanh(query + value), axis=-1)
3. Use scores to calculate a distribution with shape [batch\_size, Tq, Tv]: distribution = tf.nn.softmax(scores).
4. Use distribution to create a linear combination of value with shape batch\_size, Tq, dim]: return tf.matmul(distribution, value).

#### Args:

* **use\_scale**: If True, will create a variable to scale the attention scores.
* **causal**: Boolean. Set to True for decoder self-attention. Adds a mask such that position icannot attend to positions j > i. This prevents the flow of information from the future towards the past.

#### Call Arguments:

* **inputs**: List of the following tensors:
  + query: Query Tensor of shape [batch\_size, Tq, dim].
  + value: Value Tensor of shape [batch\_size, Tv, dim].
  + key: Optional key Tensor of shape [batch\_size, Tv, dim]. If not given, will use valuefor both key and value, which is the most common case.
* **mask**: List of the following tensors:
  + query\_mask: A boolean mask Tensor of shape [batch\_size, Tq]. If given, the output will be zero at the positions where mask==False.
  + value\_mask: A boolean mask Tensor of shape [batch\_size, Tv]. If given, will apply the mask such that values at positions where mask==False do not contribute to the result.

#### Output shape:

Attention outputs of shape [batch\_size, Tq, dim].

The meaning of query, value and key depend on the application. In the case of text similarity, for example, query is the sequence embeddings of the first piece of text and value is the sequence embeddings of the second piece of text. key is usually the same tensor as value.

Here is a code example for using AdditiveAttention in a CNN+Attention network:

# Variable-length int sequences.  
query\_input = tf.keras.Input(shape=(None,), dtype='int32')  
value\_input = tf.keras.Input(shape=(None,), dtype='int32')  
  
# Embedding lookup.  
token\_embedding = tf.keras.layers.Embedding(max\_tokens, dimension)  
# Query embeddings of shape [batch\_size, Tq, dimension].  
query\_embeddings = token\_embedding(query\_input)  
# Value embeddings of shape [batch\_size, Tv, dimension].  
value\_embeddings = token\_embedding(query\_input)  
  
# CNN layer.  
cnn\_layer = tf.keras.layers.Conv1D(  
    filters=100,  
    kernel\_size=4,  
    # Use 'same' padding so outputs have the same shape as inputs.  
    padding='same')  
# Query encoding of shape [batch\_size, Tq, filters].  
query\_seq\_encoding = cnn\_layer(query\_embeddings)  
# Value encoding of shape [batch\_size, Tv, filters].  
value\_seq\_encoding = cnn\_layer(value\_embeddings)  
  
# Query-value attention of shape [batch\_size, Tq, filters].  
query\_value\_attention\_seq = tf.keras.layers.AdditiveAttention()(  
    [query\_seq\_encoding, value\_seq\_encoding])  
  
# Reduce over the sequence axis to produce encodings of shape  
# [batch\_size, filters].  
query\_encoding = tf.keras.layers.GlobalAveragePooling1D()(  
    query\_seq\_encoding)  
query\_value\_attention = tf.keras.layers.GlobalAveragePooling1D()(  
    query\_value\_attention\_seq)  
  
# Concatenate query and document encodings to produce a DNN input layer.  
input\_layer = tf.keras.layers.Concatenate()(  
    [query\_encoding, query\_value\_attention])  
  
# Add DNN layers, and create Model.  
# ...

## \_\_init\_\_

\_\_init\_\_(  
    use\_scale=True,  
    \*\*kwargs  
)

# tf.keras.layers.AlphaDropout

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AlphaDropout#top_of_page)
* [Class AlphaDropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AlphaDropout#class_alphadropout)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AlphaDropout#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AlphaDropout#__init__)

## Class AlphaDropout

Applies Alpha Dropout to the input.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.AlphaDropout
* Class tf.compat.v2.keras.layers.AlphaDropout
* Class tf.keras.layers.AlphaDropout

Defined in [python/keras/layers/noise.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/noise.py).

Alpha Dropout is a Dropout that keeps mean and variance of inputs to their original values, in order to ensure the self-normalizing property even after this dropout. Alpha Dropout fits well to Scaled Exponential Linear Units by randomly setting activations to the negative saturation value.

#### Arguments:

* **rate**: float, drop probability (as with Dropout). The multiplicative noise will have standard deviation sqrt(rate / (1 - rate)).
* **seed**: A Python integer to use as random seed.

#### Call Arguments:

* **inputs**: Input tensor (of any rank).
* **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

## \_\_init\_\_

\_\_init\_\_(  
    rate,  
    noise\_shape=None,  
    seed=None,  
    \*\*kwargs  
)

# tf.keras.layers.Attention

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Attention#top_of_page)
* [Class Attention](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Attention#class_attention)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Attention#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Attention#__init__)

## Class Attention

Dot-product attention layer, a.k.a. Luong-style attention.

### Aliases:

* Class tf.compat.v1.keras.layers.Attention
* Class tf.compat.v2.keras.layers.Attention
* Class tf.keras.layers.Attention

Defined in [python/keras/layers/dense\_attention.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/dense_attention.py).

Inputs are query tensor of shape [batch\_size, Tq, dim], value tensor of shape [batch\_size, Tv, dim] and key tensor of shape [batch\_size, Tv, dim]. The calculation follows the steps:

1. Calculate scores with shape [batch\_size, Tq, Tv] as a query-key dot product: scores = tf.matmul(query, key, transpose\_b=True).
2. Use scores to calculate a distribution with shape [batch\_size, Tq, Tv]: distribution = tf.nn.softmax(scores).
3. Use distribution to create a linear combination of value with shape batch\_size, Tq, dim]: return tf.matmul(distribution, value).

#### Args:

* **use\_scale**: If True, will create a scalar variable to scale the attention scores.
* **causal**: Boolean. Set to True for decoder self-attention. Adds a mask such that position icannot attend to positions j > i. This prevents the flow of information from the future towards the past.

#### Call Arguments:

* **inputs**: List of the following tensors:
  + query: Query Tensor of shape [batch\_size, Tq, dim].
  + value: Value Tensor of shape [batch\_size, Tv, dim].
  + key: Optional key Tensor of shape [batch\_size, Tv, dim]. If not given, will use valuefor both key and value, which is the most common case.
* **mask**: List of the following tensors:
  + query\_mask: A boolean mask Tensor of shape [batch\_size, Tq]. If given, the output will be zero at the positions where mask==False.
  + value\_mask: A boolean mask Tensor of shape [batch\_size, Tv]. If given, will apply the mask such that values at positions where mask==False do not contribute to the result.

#### Output shape:

Attention outputs of shape [batch\_size, Tq, dim].

The meaning of query, value and key depend on the application. In the case of text similarity, for example, query is the sequence embeddings of the first piece of text and value is the sequence embeddings of the second piece of text. key is usually the same tensor as value.

Here is a code example for using Attention in a CNN+Attention network:

# Variable-length int sequences.  
query\_input = tf.keras.Input(shape=(None,), dtype='int32')  
value\_input = tf.keras.Input(shape=(None,), dtype='int32')  
  
# Embedding lookup.  
token\_embedding = tf.keras.layers.Embedding(max\_tokens, dimension)  
# Query embeddings of shape [batch\_size, Tq, dimension].  
query\_embeddings = token\_embedding(query\_input)  
# Value embeddings of shape [batch\_size, Tv, dimension].  
value\_embeddings = token\_embedding(query\_input)  
  
# CNN layer.  
cnn\_layer = tf.keras.layers.Conv1D(  
    filters=100,  
    kernel\_size=4,  
    # Use 'same' padding so outputs have the same shape as inputs.  
    padding='same')  
# Query encoding of shape [batch\_size, Tq, filters].  
query\_seq\_encoding = cnn\_layer(query\_embeddings)  
# Value encoding of shape [batch\_size, Tv, filters].  
value\_seq\_encoding = cnn\_layer(value\_embeddings)  
  
# Query-value attention of shape [batch\_size, Tq, filters].  
query\_value\_attention\_seq = tf.keras.layers.Attention()(  
    [query\_seq\_encoding, value\_seq\_encoding])  
  
# Reduce over the sequence axis to produce encodings of shape  
# [batch\_size, filters].  
query\_encoding = tf.keras.layers.GlobalAveragePooling1D()(  
    query\_seq\_encoding)  
query\_value\_attention = tf.keras.layers.GlobalAveragePooling1D()(  
    query\_value\_attention\_seq)  
  
# Concatenate query and document encodings to produce a DNN input layer.  
input\_layer = tf.keras.layers.Concatenate()(  
    [query\_encoding, query\_value\_attention])  
  
# Add DNN layers, and create Model.  
# ...

## \_\_init\_\_

\_\_init\_\_(  
    use\_scale=False,  
    \*\*kwargs  
)

tf.keras.layers.Average

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Average#top_of_page)
* [Class Average](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Average#class_average)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Average#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Average#__init__)

Class Average

Layer that averages a list of inputs.

Aliases:

* Class tf.compat.v1.keras.layers.Average
* Class tf.compat.v2.keras.layers.Average
* Class tf.keras.layers.Average

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

It takes as input a list of tensors, all of the same shape, and returns a single tensor (also of the same shape).

\_\_init\_\_

\_\_init\_\_(\*\*kwargs)

# tf.keras.layers.average

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/average#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/average#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/average#used_in_the_guide)

Functional interface to the Average layer.

### Aliases:

* tf.compat.v1.keras.layers.average
* tf.compat.v2.keras.layers.average
* tf.keras.layers.average

tf.keras.layers.average(  
    inputs,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the average of the inputs.

# tf.keras.layers.AveragePooling1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling1D#top_of_page)
* [Class AveragePooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling1D#class_averagepooling1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling1D#__init__)

## Class AveragePooling1D

Average pooling for temporal data.

### Aliases:

* Class tf.compat.v1.keras.layers.AveragePooling1D
* Class tf.compat.v1.keras.layers.AvgPool1D
* Class tf.compat.v2.keras.layers.AveragePooling1D
* Class tf.compat.v2.keras.layers.AvgPool1D
* Class tf.keras.layers.AveragePooling1D
* Class tf.keras.layers.AvgPool1D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

#### Arguments:

* **pool\_size**: Integer, size of the average pooling windows.
* **strides**: Integer, or None. Factor by which to downscale. E.g. 2 will halve the input. If None, it will default to pool\_size.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, steps, features) while channels\_first corresponds to inputs with shape (batch, features, steps).

#### Input shape:

* If data\_format='channels\_last': 3D tensor with shape (batch\_size, steps, features).
* If data\_format='channels\_first': 3D tensor with shape (batch\_size, features, steps).

#### Output shape:

* If data\_format='channels\_last': 3D tensor with shape (batch\_size, downsampled\_steps, features).
* If data\_format='channels\_first': 3D tensor with shape (batch\_size, features, downsampled\_steps).

## \_\_init\_\_

\_\_init\_\_(  
    pool\_size=2,  
    strides=None,  
    padding='valid',  
    data\_format='channels\_last',  
    \*\*kwargs  
)

# tf.keras.layers.AveragePooling2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling2D#top_of_page)
* [Class AveragePooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling2D#class_averagepooling2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling2D#__init__)

## Class AveragePooling2D

Average pooling operation for spatial data.

### Aliases:

* Class tf.compat.v1.keras.layers.AveragePooling2D
* Class tf.compat.v1.keras.layers.AvgPool2D
* Class tf.compat.v2.keras.layers.AveragePooling2D
* Class tf.compat.v2.keras.layers.AvgPool2D
* Class tf.keras.layers.AveragePooling2D
* Class tf.keras.layers.AvgPool2D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

#### Arguments:

* **pool\_size**: integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal).(2, 2) will halve the input in both spatial dimension. If only one integer is specified, the same window length will be used for both dimensions.
* **strides**: Integer, tuple of 2 integers, or None. Strides values. If None, it will default to pool\_size.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 4D tensor with shape (batch\_size, rows, cols, channels).
* If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, rows, cols).

#### Output shape:

* If data\_format='channels\_last': 4D tensor with shape (batch\_size, pooled\_rows, pooled\_cols, channels).
* If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, pooled\_rows, pooled\_cols).

## \_\_init\_\_

\_\_init\_\_(  
    pool\_size=(2, 2),  
    strides=None,  
    padding='valid',  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.AveragePooling3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling3D#top_of_page)
* [Class AveragePooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling3D#class_averagepooling3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/AveragePooling3D#__init__)

## Class AveragePooling3D

Average pooling operation for 3D data (spatial or spatio-temporal).

### Aliases:

* Class tf.compat.v1.keras.layers.AveragePooling3D
* Class tf.compat.v1.keras.layers.AvgPool3D
* Class tf.compat.v2.keras.layers.AveragePooling3D
* Class tf.compat.v2.keras.layers.AvgPool3D
* Class tf.keras.layers.AveragePooling3D
* Class tf.keras.layers.AvgPool3D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

#### Arguments:

* **pool\_size**: tuple of 3 integers, factors by which to downscale (dim1, dim2, dim3). (2, 2, 2)will halve the size of the 3D input in each dimension.
* **strides**: tuple of 3 integers, or None. Strides values.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 5D tensor with shape: (batch\_size, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels)
* If data\_format='channels\_first': 5D tensor with shape: (batch\_size, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3)

#### Output shape:

* If data\_format='channels\_last': 5D tensor with shape: (batch\_size, pooled\_dim1, pooled\_dim2, pooled\_dim3, channels)
* If data\_format='channels\_first': 5D tensor with shape: (batch\_size, channels, pooled\_dim1, pooled\_dim2, pooled\_dim3)

## \_\_init\_\_

\_\_init\_\_(  
    pool\_size=(2, 2, 2),  
    strides=None,  
    padding='valid',  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.BatchNormalization

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization#top_of_page)
* [Class BatchNormalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization#class_batchnormalization)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/BatchNormalization#__init__)

## Class BatchNormalization

Base class of Batch normalization layer (Ioffe and Szegedy, 2014).

### Aliases:

* Class tf.compat.v2.keras.layers.BatchNormalization
* Class tf.keras.layers.BatchNormalization

Defined in [python/keras/layers/normalization\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/normalization_v2.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)

### Used in the tutorials:

* [Custom layers](https://www.tensorflow.org/beta/tutorials/eager/custom_layers)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

#### Arguments:

* **axis**: Integer, the axis that should be normalized (typically the features axis). For instance, after a Conv2D layer with data\_format="channels\_first", set axis=1 in BatchNormalization.
* **momentum**: Momentum for the moving average.
* **epsilon**: Small float added to variance to avoid dividing by zero.
* **center**: If True, add offset of beta to normalized tensor. If False, beta is ignored.
* **scale**: If True, multiply by gamma. If False, gamma is not used. When the next layer is linear (also e.g. nn.relu), this can be disabled since the scaling will be done by the next layer.
* **beta\_initializer**: Initializer for the beta weight.
* **gamma\_initializer**: Initializer for the gamma weight.
* **moving\_mean\_initializer**: Initializer for the moving mean.
* **moving\_variance\_initializer**: Initializer for the moving variance.
* **beta\_regularizer**: Optional regularizer for the beta weight.
* **gamma\_regularizer**: Optional regularizer for the gamma weight.
* **beta\_constraint**: Optional constraint for the beta weight.
* **gamma\_constraint**: Optional constraint for the gamma weight.
* **renorm**: Whether to use Batch Renormalization (https://arxiv.org/abs/1702.03275). This adds extra variables during training. The inference is the same for either value of this parameter.
* **renorm\_clipping**: A dictionary that may map keys 'rmax', 'rmin', 'dmax' to scalar Tensors used to clip the renorm correction. The correction (r, d) is used as corrected\_value = normalized\_value \* r + d, with r clipped to [rmin, rmax], and d to [-dmax, dmax]. Missing rmax, rmin, dmax are set to inf, 0, inf, respectively.
* **renorm\_momentum**: Momentum used to update the moving means and standard deviations with renorm. Unlike momentum, this affects training and should be neither too small (which would add noise) nor too large (which would give stale estimates). Note that momentum is still applied to get the means and variances for inference.
* **fused**: if True, use a faster, fused implementation, or raise a ValueError if the fused implementation cannot be used. If None, use the faster implementation if possible. If False, do not used the fused implementation.
* **trainable**: Boolean, if True the variables will be marked as trainable.
* **virtual\_batch\_size**: An int. By default, virtual\_batch\_size is None, which means batch normalization is performed across the whole batch. When virtual\_batch\_size is not None, instead perform "Ghost Batch Normalization", which creates virtual sub-batches which are each normalized separately (with shared gamma, beta, and moving statistics). Must divide the actual batch size during execution.
* **adjustment**: A function taking the Tensor containing the (dynamic) shape of the input tensor and returning a pair (scale, bias) to apply to the normalized values (before gamma and beta), only during training. For example, if axis==-1, adjustment = lambda shape: ( tf.random.uniform(shape[-1:], 0.93, 1.07), tf.random.uniform(shape[-1:], -0.1, 0.1)) will scale the normalized value by up to 7% up or down, then shift the result by up to 0.1 (with independent scaling and bias for each feature but shared across all examples), and finally apply gamma and/or beta. If None, no adjustment is applied. Cannot be specified if virtual\_batch\_size is specified.

#### Call Arguments:

* **inputs**: Input tensor (of any rank).
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode.
  + training=True: The layer will normalize its inputs using the mean and variance of the current batch of inputs.
  + training=False: The layer will normalize its inputs using the mean and variance of its moving statistics, learned during training.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

#### References:

* [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift](https://arxiv.org/abs/1502.03167)

{ {TRAINABLE\_ATTRIBUTE\_NOTE}}

## \_\_init\_\_

\_\_init\_\_(  
    axis=-1,  
    momentum=0.99,  
    epsilon=0.001,  
    center=True,  
    scale=True,  
    beta\_initializer='zeros',  
    gamma\_initializer='ones',  
    moving\_mean\_initializer='zeros',  
    moving\_variance\_initializer='ones',  
    beta\_regularizer=None,  
    gamma\_regularizer=None,  
    beta\_constraint=None,  
    gamma\_constraint=None,  
    renorm=False,  
    renorm\_clipping=None,  
    renorm\_momentum=0.99,  
    fused=None,  
    trainable=True,  
    virtual\_batch\_size=None,  
    adjustment=None,  
    name=None,  
    \*\*kwargs  
)

# tf.keras.layers.Bidirectional

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Bidirectional#top_of_page)
* [Class Bidirectional](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Bidirectional#class_bidirectional)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Bidirectional#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Bidirectional#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Bidirectional#__init__)

## Class Bidirectional

Bidirectional wrapper for RNNs.

Inherits From: [Wrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper)

### Aliases:

* Class tf.compat.v1.keras.layers.Bidirectional
* Class tf.compat.v2.keras.layers.Bidirectional
* Class tf.keras.layers.Bidirectional

Defined in [python/keras/layers/wrappers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/wrappers.py).

### Used in the tutorials:

* [Load text with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/text)
* [Text classification with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_classification_rnn)

#### Arguments:

* **layer**: Recurrent instance.
* **merge\_mode**: Mode by which outputs of the forward and backward RNNs will be combined. One of {'sum', 'mul', 'concat', 'ave', None}. If None, the outputs will not be combined, they will be returned as a list.
* **backward\_layer**: Optional Recurrent instance to be used to handle backwards input processing. If backward\_layer is not provided, the layer instance passed as the layerargument will be used to generate the backward layer automatically. Note that the provided backward\_layer layer should have properties matching those of the layer argument, in particular it should have the same values for stateful, return\_states, return\_sequence, etc. In addition, backward\_layer and layer should have different go\_backwards argument values. A ValueError will be raised if these requirements are not met.

#### Call Arguments:

The call **Arguments** for this layer are the same as those of the wrapped RNN layer.

#### Raises:

* **ValueError**: 1. If layer or backward\_layer is not a Layer instance.
  1. In case of invalid merge\_mode argument.
  2. If backward\_layer has mismatched properties compared to layer.

#### Examples:

model = Sequential()  
model.add(Bidirectional(LSTM(10, return\_sequences=True), input\_shape=(5, 10)))  
model.add(Bidirectional(LSTM(10)))  
model.add(Dense(5))  
model.add(Activation('softmax'))  
model.compile(loss='categorical\_crossentropy', optimizer='rmsprop')  
  
 # With custom backward layer  
 model = Sequential()  
 forward\_layer = LSTM(10, return\_sequences=True)  
 backard\_layer = LSTM(10, activation='relu', return\_sequences=True,  
                      go\_backwards=True)  
 model.add(Bidirectional(forward\_layer, backward\_layer=backward\_layer,  
                         input\_shape=(5, 10)))  
 model.add(Dense(5))  
 model.add(Activation('softmax'))  
 model.compile(loss='categorical\_crossentropy', optimizer='rmsprop')

## \_\_init\_\_

\_\_init\_\_(  
    layer,  
    merge\_mode='concat',  
    weights=None,  
    backward\_layer=None,  
    \*\*kwargs  
)

## Properties

### constraints

## Methods

### reset\_states

reset\_states()

# tf.keras.layers.Concatenate

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Concatenate#top_of_page)
* [Class Concatenate](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Concatenate#class_concatenate)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Concatenate#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Concatenate#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Concatenate#__init__)

## Class Concatenate

Layer that concatenates a list of inputs.

### Aliases:

* Class tf.compat.v1.keras.layers.Concatenate
* Class tf.compat.v2.keras.layers.Concatenate
* Class tf.keras.layers.Concatenate

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

It takes as input a list of tensors, all of the same shape except for the concatenation axis, and returns a single tensor, the concatenation of all inputs.

#### Arguments:

* **axis**: Axis along which to concatenate.
* **\*\*kwargs**: standard layer keyword **Arguments**.

## \_\_init\_\_

\_\_init\_\_(  
    axis=-1,  
    \*\*kwargs  
)

# tf.keras.layers.concatenate

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/concatenate#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/concatenate#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/concatenate#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/concatenate#used_in_the_tutorials)

Functional interface to the Concatenate layer.

### Aliases:

* tf.compat.v1.keras.layers.concatenate
* tf.compat.v2.keras.layers.concatenate
* tf.keras.layers.concatenate

tf.keras.layers.concatenate(  
    inputs,  
    axis=-1,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **axis**: Concatenation axis.
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the concatenation of the inputs alongside axis axis.

# tf.keras.layers.Conv1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D#top_of_page)
* [Class Conv1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D#class_conv1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv1D#__init__)

## Class Conv1D

1D convolution layer (e.g. temporal convolution).

### Aliases:

* Class tf.compat.v1.keras.layers.Conv1D
* Class tf.compat.v1.keras.layers.Convolution1D
* Class tf.compat.v2.keras.layers.Conv1D
* Class tf.compat.v2.keras.layers.Convolution1D
* Class tf.keras.layers.Conv1D
* Class tf.keras.layers.Convolution1D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide an input\_shape argument (tuple of integers or None, e.g. (10, 128) for sequences of 10 vectors of 128-dimensional vectors, or (None, 128) for variable-length sequences of 128-dimensional vectors.

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of a single integer, specifying the length of the 1D convolution window.
* **strides**: An integer or tuple/list of a single integer, specifying the stride length of the convolution. Specifying any stride value != 1 is incompatible with specifying any dilation\_ratevalue != 1.
* **padding**: One of "valid", "causal" or "same" (case-insensitive). "causal" results in causal (dilated) convolutions, e.g. output[t] does not depend on input[t+1:]. Useful when modeling temporal data where the model should not violate the temporal order. See [WaveNet: A Generative Model for Raw Audio, section 2.1](https://arxiv.org/abs/1609.03499).
* **data\_format**: A string, one of channels\_last (default) or channels\_first.
* **dilation\_rate**: an integer or tuple/list of a single integer, specifying the dilation rate to use for dilated convolution. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any strides value != 1.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

3D tensor with shape: (batch\_size, steps, input\_dim)

#### Output shape:

3D tensor with shape: (batch\_size, new\_steps, filters) steps value might have changed due to padding or strides.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=1,  
    padding='valid',  
    data\_format='channels\_last',  
    dilation\_rate=1,  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.Conv2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D#top_of_page)
* [Class Conv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D#class_conv2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D#__init__)

## Class Conv2D

2D convolution layer (e.g. spatial convolution over images).

### Aliases:

* Class tf.compat.v1.keras.layers.Conv2D
* Class tf.compat.v1.keras.layers.Convolution2D
* Class tf.compat.v2.keras.layers.Conv2D
* Class tf.compat.v2.keras.layers.Convolution2D
* Class tf.keras.layers.Conv2D
* Class tf.keras.layers.Convolution2D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Convolutional Neural Networks](https://www.tensorflow.org/beta/tutorials/images/intro_to_cnns)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom layers](https://www.tensorflow.org/beta/tutorials/eager/custom_layers)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Multi-worker Training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_keras)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activationis not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers, does not include the sample axis), e.g. input\_shape=(128, 128, 3) for 128x128 RGB pictures in data\_format="channels\_last".

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: one of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **dilation\_rate**: an integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any stride value != 1.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

4D tensor with shape: (samples, channels, rows, cols) if data\_format='channels\_first' or 4D tensor with shape: (samples, rows, cols, channels) if data\_format='channels\_last'.

#### Output shape:

4D tensor with shape: (samples, filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4D tensor with shape: (samples, new\_rows, new\_cols, filters) if data\_format='channels\_last'. rows and cols values might have changed due to padding.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1),  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.Conv2DTranspose

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose#top_of_page)
* [Class Conv2DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose#class_conv2dtranspose)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2DTranspose#__init__)

## Class Conv2DTranspose

Transposed convolution layer (sometimes called Deconvolution).

Inherits From: [Conv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D)

### Aliases:

* Class tf.compat.v1.keras.layers.Conv2DTranspose
* Class tf.compat.v1.keras.layers.Convolution2DTranspose
* Class tf.compat.v2.keras.layers.Conv2DTranspose
* Class tf.compat.v2.keras.layers.Convolution2DTranspose
* Class tf.keras.layers.Conv2DTranspose
* Class tf.keras.layers.Convolution2DTranspose

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

The need for transposed convolutions generally arises from the desire to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution.

When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers, does not include the sample axis), e.g. input\_shape=(128, 128, 3) for 128x128 RGB pictures in data\_format="channels\_last".

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: one of "valid" or "same" (case-insensitive).
* **output\_padding**: An integer or tuple/list of 2 integers, specifying the amount of padding along the height and width of the output tensor. Can be a single integer to specify the same value for all spatial dimensions. The amount of output padding along a given dimension must be lower than the stride along that same dimension. If set to None (default), the output shape is inferred.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **dilation\_rate**: an integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any stride value != 1.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

4D tensor with shape: (batch, channels, rows, cols) if data\_format='channels\_first' or 4D tensor with shape: (batch, rows, cols, channels) if data\_format='channels\_last'.

#### Output shape:

4D tensor with shape: (batch, filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4D tensor with shape: (batch, new\_rows, new\_cols, filters) if data\_format='channels\_last'.rows and cols values might have changed due to padding.

#### References:

* [A guide to convolution arithmetic for deep learning](https://arxiv.org/abs/1603.07285v1)
* [Deconvolutional Networks](https://www.matthewzeiler.com/mattzeiler/deconvolutionalnetworks.pdf)

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1),  
    padding='valid',  
    output\_padding=None,  
    data\_format=None,  
    dilation\_rate=(1, 1),  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.Conv3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D#top_of_page)
* [Class Conv3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D#class_conv3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D#__init__)

## Class Conv3D

3D convolution layer (e.g. spatial convolution over volumes).

### Aliases:

* Class tf.compat.v1.keras.layers.Conv3D
* Class tf.compat.v1.keras.layers.Convolution3D
* Class tf.compat.v2.keras.layers.Conv3D
* Class tf.compat.v2.keras.layers.Convolution3D
* Class tf.keras.layers.Conv3D
* Class tf.keras.layers.Convolution3D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activationis not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers, does not include the sample axis), e.g. input\_shape=(128, 128, 128, 1) for 128x128x128 volumes with a single channel, in data\_format="channels\_last".

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 3 integers, specifying the depth, height and width of the 3D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 3 integers, specifying the strides of the convolution along each spatial dimension. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_ratevalue != 1.
* **padding**: one of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **dilation\_rate**: an integer or tuple/list of 3 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any stride value != 1.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

5D tensor with shape: (samples, channels, conv\_dim1, conv\_dim2, conv\_dim3) if data\_format='channels\_first' or 5D tensor with shape: (samples, conv\_dim1, conv\_dim2, conv\_dim3, channels) if data\_format='channels\_last'.

#### Output shape:

5D tensor with shape: (samples, filters, new\_conv\_dim1, new\_conv\_dim2, new\_conv\_dim3) if data\_format='channels\_first' or 5D tensor with shape: (samples, new\_conv\_dim1, new\_conv\_dim2, new\_conv\_dim3, filters) if data\_format='channels\_last'. new\_conv\_dim1, new\_conv\_dim2 and new\_conv\_dim3 values might have changed due to padding.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1, 1),  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.Conv3DTranspose

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3DTranspose#top_of_page)
* [Class Conv3DTranspose](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3DTranspose#class_conv3dtranspose)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3DTranspose#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3DTranspose#__init__)

## Class Conv3DTranspose

Transposed convolution layer (sometimes called Deconvolution).

Inherits From: [Conv3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv3D)

### Aliases:

* Class tf.compat.v1.keras.layers.Conv3DTranspose
* Class tf.compat.v1.keras.layers.Convolution3DTranspose
* Class tf.compat.v2.keras.layers.Conv3DTranspose
* Class tf.compat.v2.keras.layers.Convolution3DTranspose
* Class tf.keras.layers.Conv3DTranspose
* Class tf.keras.layers.Convolution3DTranspose

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

The need for transposed convolutions generally arises from the desire to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution.

When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers, does not include the sample axis), e.g. input\_shape=(128, 128, 128, 3) for a 128x128x128 volume with 3 channels if data\_format="channels\_last".

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 3 integers, specifying the depth, height and width of the 3D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 3 integers, specifying the strides of the convolution along the depth, height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_ratevalue != 1.
* **padding**: one of "valid" or "same" (case-insensitive).
* **output\_padding**: An integer or tuple/list of 3 integers, specifying the amount of padding along the depth, height, and width. Can be a single integer to specify the same value for all spatial dimensions. The amount of output padding along a given dimension must be lower than the stride along that same dimension. If set to None (default), the output shape is inferred.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, depth, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, depth, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **dilation\_rate**: an integer or tuple/list of 3 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any stride value != 1.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

5D tensor with shape: (batch, channels, depth, rows, cols) if data\_format='channels\_first' or 5D tensor with shape: (batch, depth, rows, cols, channels) if data\_format='channels\_last'.

#### Output shape:

5D tensor with shape: (batch, filters, new\_depth, new\_rows, new\_cols) if data\_format='channels\_first' or 5D tensor with shape: (batch, new\_depth, new\_rows, new\_cols, filters) if data\_format='channels\_last'. depth and rows and cols values might have changed due to padding.

#### References:

* [A guide to convolution arithmetic for deep learning](https://arxiv.org/abs/1603.07285v1)
* [Deconvolutional Networks](https://www.matthewzeiler.com/mattzeiler/deconvolutionalnetworks.pdf)

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1, 1),  
    padding='valid',  
    output\_padding=None,  
    data\_format=None,  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.ConvLSTM2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ConvLSTM2D#top_of_page)
* [Class ConvLSTM2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ConvLSTM2D#class_convlstm2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ConvLSTM2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ConvLSTM2D#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ConvLSTM2D#properties)

## Class ConvLSTM2D

Convolutional LSTM.

### Aliases:

* Class tf.compat.v1.keras.layers.ConvLSTM2D
* Class tf.compat.v2.keras.layers.ConvLSTM2D
* Class tf.keras.layers.ConvLSTM2D

Defined in [python/keras/layers/convolutional\_recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional_recurrent.py).

It is similar to an LSTM layer, but the input transformations and recurrent transformations are both convolutional.

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of n integers, specifying the dimensions of the convolution window.
* **strides**: An integer or tuple/list of n integers, specifying the strides of the convolution. Specifying any stride value != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, time, ..., channels) while channels\_first corresponds to inputs with shape (batch, time, channels, ...). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **dilation\_rate**: An integer or tuple/list of n integers, specifying the dilation rate to use for dilated convolution. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any strides value != 1.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **recurrent\_activation**: Activation function to use for the recurrent step.
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.
* **bias\_initializer**: Initializer for the bias vector.
* **unit\_forget\_bias**: Boolean. If True, add 1 to the bias of the forget gate at initialization. Use in combination with bias\_initializer="zeros". This is recommended in [Jozefowicz et al.](http://www.jmlr.org/proceedings/papers/v37/jozefowicz15.pdf)
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to.
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **return\_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence.
* **go\_backwards**: Boolean (default False). If True, process the input sequence backwards.
* **stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.

#### Call Arguments:

* **inputs**: A 5D tensor.
* **mask**: Binary tensor of shape (samples, timesteps) indicating whether a given timestep should be masked.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. This argument is passed to the cell when calling it. This is only relevant if dropout or recurrent\_dropout are set.
* **initial\_state**: List of initial state tensors to be passed to the first call of the cell.

#### Input shape:

* If data\_format='channels\_first' 5D tensor with shape: (samples, time, channels, rows, cols)
* If data\_format='channels\_last' 5D tensor with shape: (samples, time, rows, cols, channels)

#### Output shape:

* If return\_sequences
  + If data\_format='channels\_first' 5D tensor with shape: (samples, time, filters, output\_row, output\_col)
  + If data\_format='channels\_last' 5D tensor with shape: (samples, time, output\_row, output\_col, filters)
* Else
  + If data\_format ='channels\_first' 4D tensor with shape: (samples, filters, output\_row, output\_col)
  + If data\_format='channels\_last' 4D tensor with shape: (samples, output\_row, output\_col, filters) where o\_row and o\_col depend on the shape of the filter and the padding

#### Raises:

* **ValueError**: in case of invalid constructor **Arguments**.

#### References:

* [Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting](http://arxiv.org/abs/1506.04214v1) The current implementation does not include the feedback loop on the cells output.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1),  
    activation='tanh',  
    recurrent\_activation='hard\_sigmoid',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    unit\_forget\_bias=True,  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    return\_sequences=False,  
    go\_backwards=False,  
    stateful=False,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    \*\*kwargs  
)

## Properties

### activation

### bias\_constraint

### bias\_initializer

### bias\_regularizer

### data\_format

### dilation\_rate

### dropout

### filters

### kernel\_constraint

### kernel\_initializer

### kernel\_regularizer

### kernel\_size

### padding

### recurrent\_activation

### recurrent\_constraint

### recurrent\_dropout

### recurrent\_initializer

### recurrent\_regularizer

### states

### strides

### unit\_forget\_bias

### use\_bias

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states

reset\_states(states=None)

# tf.keras.layers.Cropping1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping1D#top_of_page)
* [Class Cropping1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping1D#class_cropping1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping1D#__init__)

## Class Cropping1D

Cropping layer for 1D input (e.g. temporal sequence).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Cropping1D
* Class tf.compat.v2.keras.layers.Cropping1D
* Class tf.keras.layers.Cropping1D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

It crops along the time dimension (axis 1).

#### Arguments:

* **cropping**: Int or tuple of int (length 2) How many units should be trimmed off at the beginning and end of the cropping dimension (axis 1). If a single int is provided, the same value will be used for both.

#### Input shape:

3D tensor with shape (batch, axis\_to\_crop, features)

#### Output shape:

3D tensor with shape (batch, cropped\_axis, features)

## \_\_init\_\_

\_\_init\_\_(  
    cropping=(1, 1),  
    \*\*kwargs  
)

# tf.keras.layers.Cropping2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping2D#top_of_page)
* [Class Cropping2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping2D#class_cropping2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping2D#__init__)

## Class Cropping2D

Cropping layer for 2D input (e.g. picture).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Cropping2D
* Class tf.compat.v2.keras.layers.Cropping2D
* Class tf.keras.layers.Cropping2D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

It crops along spatial dimensions, i.e. height and width.

#### Arguments:

* **cropping**: Int, or tuple of 2 ints, or tuple of 2 tuples of 2 ints.
  + If int: the same symmetric cropping is applied to height and width.
  + If tuple of 2 ints: interpreted as two different symmetric cropping values for height and width: (symmetric\_height\_crop, symmetric\_width\_crop).
  + If tuple of 2 tuples of 2 ints: interpreted as ((top\_crop, bottom\_crop), (left\_crop, right\_crop))
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

4D tensor with shape: - If data\_format is "channels\_last": (batch, rows, cols, channels) - If data\_format is "channels\_first": (batch, channels, rows, cols)

#### Output shape:

4D tensor with shape: - If data\_format is "channels\_last": (batch, cropped\_rows, cropped\_cols, channels) - If data\_format is "channels\_first": (batch, channels, cropped\_rows, cropped\_cols)

#### Examples:

# Crop the input 2D images or feature maps  
model = Sequential()  
model.add(Cropping2D(cropping=((2, 2), (4, 4)),  
                     input\_shape=(28, 28, 3)))  
# now model.output\_shape == (None, 24, 20, 3)  
model.add(Conv2D(64, (3, 3), padding='same'))  
model.add(Cropping2D(cropping=((2, 2), (2, 2))))  
# now model.output\_shape == (None, 20, 16. 64)

## \_\_init\_\_

\_\_init\_\_(  
    cropping=((0, 0), (0, 0)),  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.Cropping3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping3D#top_of_page)
* [Class Cropping3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping3D#class_cropping3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Cropping3D#__init__)

## Class Cropping3D

Cropping layer for 3D data (e.g. spatial or spatio-temporal).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Cropping3D
* Class tf.compat.v2.keras.layers.Cropping3D
* Class tf.keras.layers.Cropping3D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

#### Arguments:

* **cropping**: Int, or tuple of 3 ints, or tuple of 3 tuples of 2 ints.
  + If int: the same symmetric cropping is applied to depth, height, and width.
  + If tuple of 3 ints: interpreted as two different symmetric cropping values for depth, height, and width: (symmetric\_dim1\_crop, symmetric\_dim2\_crop, symmetric\_dim3\_crop).
  + If tuple of 3 tuples of 2 ints: interpreted as ((left\_dim1\_crop, right\_dim1\_crop), (left\_dim2\_crop, right\_dim2\_crop), (left\_dim3\_crop, right\_dim3\_crop))
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

5D tensor with shape: - If data\_format is "channels\_last": (batch, first\_axis\_to\_crop, second\_axis\_to\_crop, third\_axis\_to\_crop, depth) - If data\_format is "channels\_first":(batch, depth, first\_axis\_to\_crop, second\_axis\_to\_crop, third\_axis\_to\_crop)

#### Output shape:

5D tensor with shape: - If data\_format is "channels\_last": (batch, first\_cropped\_axis, second\_cropped\_axis, third\_cropped\_axis, depth) - If data\_format is "channels\_first":(batch, depth, first\_cropped\_axis, second\_cropped\_axis, third\_cropped\_axis)

## \_\_init\_\_

\_\_init\_\_(  
    cropping=((1, 1), (1, 1), (1, 1)),  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.Dense

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense#top_of_page)
* [Class Dense](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense#class_dense)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense#__init__)

## Class Dense

Just your regular densely-connected NN layer.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Dense
* Class tf.compat.v2.keras.layers.Dense
* Class tf.keras.layers.Dense

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Keras custom callbacks](https://www.tensorflow.org/beta/guide/keras/custom_callback)
* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)
* [Using GPUs](https://www.tensorflow.org/beta/guide/using_gpu)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)
* [Convolutional Neural Networks](https://www.tensorflow.org/beta/tutorials/images/intro_to_cnns)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom layers](https://www.tensorflow.org/beta/tutorials/eager/custom_layers)
* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Explore overfitting and underfitting](https://www.tensorflow.org/beta/tutorials/keras/overfit_and_underfit)
* [Get started with TensorFlow 2.0 for beginners](https://www.tensorflow.org/beta/tutorials/quickstart/beginner)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load CSV with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/csv)
* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Load text with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/text)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Multi-worker Training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_keras)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Regression: Predict fuel efficiency](https://www.tensorflow.org/beta/tutorials/keras/basic_regression)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)
* [Text classification of movie reviews with Keras and TensorFlow Hub](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification_with_tfhub)
* [Text classification with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_classification_rnn)
* [Text classification with movie reviews](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Train your first neural network: basic classification](https://www.tensorflow.org/beta/tutorials/keras/basic_classification)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [Word embeddings](https://www.tensorflow.org/beta/tutorials/text/word_embeddings)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernelis a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True).

**Note:** If the input to the layer has a rank greater than 2, then it is flattened prior to the initial dot product with **kernel**.

#### Example:

# as first layer in a sequential model:  
model = Sequential()  
model.add(Dense(32, input\_shape=(16,)))  
# now the model will take as input arrays of shape (\*, 16)  
# and output arrays of shape (\*, 32)  
  
# after the first layer, you don't need to specify  
# the size of the input anymore:  
model.add(Dense(32))

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

N-D tensor with shape: (batch\_size, ..., input\_dim). The most common situation would be a 2D input with shape (batch\_size, input\_dim).

#### Output shape:

N-D tensor with shape: (batch\_size, ..., units). For instance, for a 2D input with shape (batch\_size, input\_dim), the output would have shape (batch\_size, units).

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.DenseFeatures

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DenseFeatures#top_of_page)
* [Class DenseFeatures](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DenseFeatures#class_densefeatures)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DenseFeatures#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DenseFeatures#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DenseFeatures#__init__)

## Class DenseFeatures

A layer that produces a dense Tensor based on given feature\_columns.

### Aliases:

* Class tf.compat.v1.keras.layers.DenseFeatures
* Class tf.compat.v2.keras.layers.DenseFeatures
* Class tf.keras.layers.DenseFeatures

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Build a linear model with Estimators](https://www.tensorflow.org/beta/tutorials/estimators/linear)
* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)

Generally a single example in training data is described with FeatureColumns. At the first layer of the model, this column oriented data should be converted to a single Tensor.

This layer can be called multiple times with different features.

#### Example:

price = numeric\_column('price')  
keywords\_embedded = embedding\_column(  
    categorical\_column\_with\_hash\_bucket("keywords", 10K), dimensions=16)  
columns = [price, keywords\_embedded, ...]  
feature\_layer = DenseFeatures(columns)  
  
features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))  
dense\_tensor = feature\_layer(features)  
for units in [128, 64, 32]:  
  dense\_tensor = tf.compat.v1.layers.dense(dense\_tensor, units, tf.nn.relu)  
prediction = tf.compat.v1.layers.dense(dense\_tensor, 1).

## \_\_init\_\_

\_\_init\_\_(  
    feature\_columns,  
    trainable=True,  
    name=None,  
    \*\*kwargs  
)

Constructs a DenseFeatures.

#### Args:

* **feature\_columns**: An iterable containing the FeatureColumns to use as inputs to your model. All items should be instances of classes derived from DenseColumn such as numeric\_column, embedding\_column, bucketized\_column, indicator\_column. If you have categorical features, you can wrap them with an embedding\_column or indicator\_column.
* **trainable**: Boolean, whether the layer's variables will be updated via gradient descent during training.
* **name**: Name to give to the DenseFeatures.
* **\*\*kwargs**: Keyword **Arguments** to construct a layer.

#### Raises:

* **ValueError**: if an item in feature\_columns is not a DenseColumn.

# tf.keras.layers.DepthwiseConv2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DepthwiseConv2D#top_of_page)
* [Class DepthwiseConv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DepthwiseConv2D#class_depthwiseconv2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DepthwiseConv2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/DepthwiseConv2D#__init__)

## Class DepthwiseConv2D

Depthwise separable 2D convolution.

Inherits From: [Conv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Conv2D)

### Aliases:

* Class tf.compat.v1.keras.layers.DepthwiseConv2D
* Class tf.compat.v2.keras.layers.DepthwiseConv2D
* Class tf.keras.layers.DepthwiseConv2D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

Depthwise Separable convolutions consists in performing just the first step in a depthwise spatial convolution (which acts on each input channel separately). The depth\_multiplier argument controls how many output channels are generated per input channel in the depthwise step.

#### Arguments:

* **kernel\_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: one of 'valid' or 'same' (case-insensitive).
* **depth\_multiplier**: The number of depthwise convolution output channels for each input channel. The total number of depthwise convolution output channels will be equal to filters\_in \* depth\_multiplier.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be 'channels\_last'.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. 'linear' activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **depthwise\_initializer**: Initializer for the depthwise kernel matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **depthwise\_regularizer**: Regularizer function applied to the depthwise kernel matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its 'activation').
* **depthwise\_constraint**: Constraint function applied to the depthwise kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

4D tensor with shape: [batch, channels, rows, cols] if data\_format='channels\_first' or 4D tensor with shape: [batch, rows, cols, channels] if data\_format='channels\_last'.

#### Output shape:

4D tensor with shape: [batch, filters, new\_rows, new\_cols] if data\_format='channels\_first' or 4D tensor with shape: [batch, new\_rows, new\_cols, filters] if data\_format='channels\_last'.rows and cols values might have changed due to padding.

## \_\_init\_\_

\_\_init\_\_(  
    kernel\_size,  
    strides=(1, 1),  
    padding='valid',  
    depth\_multiplier=1,  
    data\_format=None,  
    activation=None,  
    use\_bias=True,  
    depthwise\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    depthwise\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    depthwise\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/deserialize#aliases)

Instantiates a layer from a config dictionary.

### Aliases:

* tf.compat.v1.keras.layers.deserialize
* tf.compat.v2.keras.layers.deserialize
* tf.keras.layers.deserialize

tf.keras.layers.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/layers/serialization.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/serialization.py).

#### Arguments:

* **config**: dict of the form {'class\_name': str, 'config': dict}
* **custom\_objects**: dict mapping class names (or function names) of custom (non-Keras) objects to class/functions

#### Returns:

Layer instance (may be Model, Sequential, Network, Layer...)

# tf.keras.layers.Dot

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dot#top_of_page)
* [Class Dot](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dot#class_dot)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dot#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dot#__init__)

## Class Dot

Layer that computes a dot product between samples in two tensors.

### Aliases:

* Class tf.compat.v1.keras.layers.Dot
* Class tf.compat.v2.keras.layers.Dot
* Class tf.keras.layers.Dot

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

E.g. if applied to a list of two tensors a and b of shape (batch\_size, n), the output will be a tensor of shape (batch\_size, 1) where each entry i will be the dot product between a[i] and b[i].

#### Arguments:

* **axes**: Integer or tuple of integers, axis or axes along which to take the dot product.
* **normalize**: Whether to L2-normalize samples along the dot product axis before taking the dot product. If set to True, then the output of the dot product is the cosine proximity between the two samples.
* **\*\*kwargs**: Standard layer keyword **Arguments**.

## \_\_init\_\_

\_\_init\_\_(  
    axes,  
    normalize=False,  
    \*\*kwargs  
)

# tf.keras.layers.dot

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/dot#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/dot#aliases)

Functional interface to the Dot layer.

### Aliases:

* tf.compat.v1.keras.layers.dot
* tf.compat.v2.keras.layers.dot
* tf.keras.layers.dot

tf.keras.layers.dot(  
    inputs,  
    axes,  
    normalize=False,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **axes**: Integer or tuple of integers, axis or axes along which to take the dot product.
* **normalize**: Whether to L2-normalize samples along the dot product axis before taking the dot product. If set to True, then the output of the dot product is the cosine proximity between the two samples.
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the dot product of the samples from the inputs.

# tf.keras.layers.Dropout

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout#top_of_page)
* [Class Dropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout#class_dropout)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout#__init__)

## Class Dropout

Applies Dropout to the input.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Dropout
* Class tf.compat.v2.keras.layers.Dropout
* Class tf.keras.layers.Dropout

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Explore overfitting and underfitting](https://www.tensorflow.org/beta/tutorials/keras/overfit_and_underfit)
* [Get started with TensorFlow 2.0 for beginners](https://www.tensorflow.org/beta/tutorials/quickstart/beginner)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

#### Arguments:

* **rate**: Float between 0 and 1. Fraction of the input units to drop.
* **noise\_shape**: 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape (batch\_size, timesteps, features) and you want the dropout mask to be the same for all timesteps, you can use noise\_shape=(batch\_size, 1, features).
* **seed**: A Python integer to use as random seed.

#### Call Arguments:

* **inputs**: Input tensor (of any rank).
* **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

## \_\_init\_\_

\_\_init\_\_(  
    rate,  
    noise\_shape=None,  
    seed=None,  
    \*\*kwargs  
)

# tf.keras.layers.ELU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ELU#top_of_page)
* [Class ELU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ELU#class_elu)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ELU#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ELU#__init__)

## Class ELU

Exponential Linear Unit.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ELU
* Class tf.compat.v2.keras.layers.ELU
* Class tf.keras.layers.ELU

Defined in [python/keras/layers/advanced\_activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/advanced_activations.py).

#### It follows:

f(x) = alpha \* (exp(x) - 1.) for x < 0, f(x) = x for x >= 0.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as the input.

#### Arguments:

* **alpha**: Scale for the negative factor.

## \_\_init\_\_

\_\_init\_\_(  
    alpha=1.0,  
    \*\*kwargs  
)

# tf.keras.layers.Embedding

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding#top_of_page)
* [Class Embedding](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding#class_embedding)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Embedding#__init__)

## Class Embedding

Turns positive integers (indexes) into dense vectors of fixed size.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Embedding
* Class tf.compat.v2.keras.layers.Embedding
* Class tf.keras.layers.Embedding

Defined in [python/keras/layers/embeddings.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/embeddings.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load text with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/text)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Text classification with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_classification_rnn)
* [Text classification with movie reviews](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [Word embeddings](https://www.tensorflow.org/beta/tutorials/text/word_embeddings)

e.g. [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]

This layer can only be used as the first layer in a model.

#### Example:

model = Sequential()  
model.add(Embedding(1000, 64, input\_length=10))  
# the model will take as input an integer matrix of size (batch,  
# input\_length).  
# the largest integer (i.e. word index) in the input should be no larger  
# than 999 (vocabulary size).  
# now model.output\_shape == (None, 10, 64), where None is the batch  
# dimension.  
  
input\_array = np.random.randint(1000, size=(32, 10))  
  
model.compile('rmsprop', 'mse')  
output\_array = model.predict(input\_array)  
assert output\_array.shape == (32, 10, 64)

#### Arguments:

* **input\_dim**: int > 0. Size of the vocabulary, i.e. maximum integer index + 1.
* **output\_dim**: int >= 0. Dimension of the dense embedding.
* **embeddings\_initializer**: Initializer for the embeddings matrix.
* **embeddings\_regularizer**: Regularizer function applied to the embeddings matrix.
* **embeddings\_constraint**: Constraint function applied to the embeddings matrix.
* **mask\_zero**: Whether or not the input value 0 is a special "padding" value that should be masked out. This is useful when using recurrent layers which may take variable length input. If this is True then all subsequent layers in the model need to support masking or an exception will be raised. If mask\_zero is set to True, as a consequence, index 0 cannot be used in the vocabulary (input\_dim should equal size of vocabulary + 1).
* **input\_length**: Length of input sequences, when it is constant. This argument is required if you are going to connect Flatten then Dense layers upstream (without it, the shape of the dense outputs cannot be computed).

#### Input shape:

2D tensor with shape: (batch\_size, input\_length).

#### Output shape:

3D tensor with shape: (batch\_size, input\_length, output\_dim).

## \_\_init\_\_

\_\_init\_\_(  
    input\_dim,  
    output\_dim,  
    embeddings\_initializer='uniform',  
    embeddings\_regularizer=None,  
    activity\_regularizer=None,  
    embeddings\_constraint=None,  
    mask\_zero=False,  
    input\_length=None,  
    \*\*kwargs  
)

# tf.keras.layers.Flatten

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten#top_of_page)
* [Class Flatten](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten#class_flatten)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Flatten#__init__)

## Class Flatten

Flattens the input. Does not affect the batch size.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Flatten
* Class tf.compat.v2.keras.layers.Flatten
* Class tf.keras.layers.Flatten

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)

### Used in the tutorials:

* [Convolutional Neural Networks](https://www.tensorflow.org/beta/tutorials/images/intro_to_cnns)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Get started with TensorFlow 2.0 for beginners](https://www.tensorflow.org/beta/tutorials/quickstart/beginner)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Multi-worker Training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_keras)
* [Train your first neural network: basic classification](https://www.tensorflow.org/beta/tutorials/keras/basic_classification)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

If inputs are shaped (batch,) without a channel dimension, then flattening adds an extra channel dimension and output shapes are (batch, 1).

#### Arguments:

* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, ..., channels) while channels\_first corresponds to inputs with shape (batch, channels, ...). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Example:

model = Sequential()  
model.add(Convolution2D(64, 3, 3,  
                        border\_mode='same',  
                        input\_shape=(3, 32, 32)))  
# now: model.output\_shape == (None, 64, 32, 32)  
  
model.add(Flatten())  
# now: model.output\_shape == (None, 65536)

## \_\_init\_\_

\_\_init\_\_(  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.GaussianDropout

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianDropout#top_of_page)
* [Class GaussianDropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianDropout#class_gaussiandropout)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianDropout#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianDropout#__init__)

## Class GaussianDropout

Apply multiplicative 1-centered Gaussian noise.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.GaussianDropout
* Class tf.compat.v2.keras.layers.GaussianDropout
* Class tf.keras.layers.GaussianDropout

Defined in [python/keras/layers/noise.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/noise.py).

As it is a regularization layer, it is only active at training time.

#### Arguments:

* **rate**: Float, drop probability (as with Dropout). The multiplicative noise will have standard deviation sqrt(rate / (1 - rate)).

#### Call Arguments:

* **inputs**: Input tensor (of any rank).
* **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

## \_\_init\_\_

\_\_init\_\_(  
    rate,  
    \*\*kwargs  
)

# tf.keras.layers.GaussianNoise

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianNoise#top_of_page)
* [Class GaussianNoise](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianNoise#class_gaussiannoise)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianNoise#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GaussianNoise#__init__)

## Class GaussianNoise

Apply additive zero-centered Gaussian noise.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.GaussianNoise
* Class tf.compat.v2.keras.layers.GaussianNoise
* Class tf.keras.layers.GaussianNoise

Defined in [python/keras/layers/noise.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/noise.py).

This is useful to mitigate overfitting (you could see it as a form of random data augmentation). Gaussian Noise (GS) is a natural choice as corruption process for real valued inputs.

As it is a regularization layer, it is only active at training time.

#### Arguments:

* **stddev**: Float, standard deviation of the noise distribution.

#### Call Arguments:

* **inputs**: Input tensor (of any rank).
* **training**: Python boolean indicating whether the layer should behave in training mode (adding noise) or in inference mode (doing nothing).

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

## \_\_init\_\_

\_\_init\_\_(  
    stddev,  
    \*\*kwargs  
)

# tf.keras.layers.GlobalAveragePooling1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D#top_of_page)
* [Class GlobalAveragePooling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D#class_globalaveragepooling1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling1D#__init__)

## Class GlobalAveragePooling1D

Global average pooling operation for temporal data.

### Aliases:

* Class tf.compat.v1.keras.layers.GlobalAveragePooling1D
* Class tf.compat.v1.keras.layers.GlobalAvgPool1D
* Class tf.compat.v2.keras.layers.GlobalAveragePooling1D
* Class tf.compat.v2.keras.layers.GlobalAvgPool1D
* Class tf.keras.layers.GlobalAveragePooling1D
* Class tf.keras.layers.GlobalAvgPool1D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Text classification with movie reviews](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification)
* [Word embeddings](https://www.tensorflow.org/beta/tutorials/text/word_embeddings)

#### Arguments:

* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, steps, features) while channels\_first corresponds to inputs with shape (batch, features, steps).

#### Call Arguments:

* **inputs**: A 3D tensor.
* **mask**: Binary tensor of shape (batch\_size, steps) indicating whether a given step should be masked (excluded from the average).

#### Input shape:

* If data\_format='channels\_last': 3D tensor with shape: (batch\_size, steps, features)
* If data\_format='channels\_first': 3D tensor with shape: (batch\_size, features, steps)

#### Output shape:

2D tensor with shape (batch\_size, features).

## \_\_init\_\_

\_\_init\_\_(  
    data\_format='channels\_last',  
    \*\*kwargs  
)

# tf.keras.layers.GlobalAveragePooling2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D#top_of_page)
* [Class GlobalAveragePooling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D#class_globalaveragepooling2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling2D#__init__)

## Class GlobalAveragePooling2D

Global average pooling operation for spatial data.

### Aliases:

* Class tf.compat.v1.keras.layers.GlobalAveragePooling2D
* Class tf.compat.v1.keras.layers.GlobalAvgPool2D
* Class tf.compat.v2.keras.layers.GlobalAveragePooling2D
* Class tf.compat.v2.keras.layers.GlobalAvgPool2D
* Class tf.keras.layers.GlobalAveragePooling2D
* Class tf.keras.layers.GlobalAvgPool2D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Transfer Learning Using Pretrained ConvNets](https://www.tensorflow.org/beta/tutorials/images/transfer_learning)

#### Arguments:

* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 4D tensor with shape (batch\_size, rows, cols, channels).
* If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, rows, cols).

#### Output shape:

2D tensor with shape (batch\_size, channels).

## \_\_init\_\_

\_\_init\_\_(  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.GlobalAveragePooling3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling3D#top_of_page)
* [Class GlobalAveragePooling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling3D#class_globalaveragepooling3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalAveragePooling3D#__init__)

## Class GlobalAveragePooling3D

Global Average pooling operation for 3D data.

### Aliases:

* Class tf.compat.v1.keras.layers.GlobalAveragePooling3D
* Class tf.compat.v1.keras.layers.GlobalAvgPool3D
* Class tf.compat.v2.keras.layers.GlobalAveragePooling3D
* Class tf.compat.v2.keras.layers.GlobalAvgPool3D
* Class tf.keras.layers.GlobalAveragePooling3D
* Class tf.keras.layers.GlobalAvgPool3D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

#### Arguments:

* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 5D tensor with shape: (batch\_size, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels)
* If data\_format='channels\_first': 5D tensor with shape: (batch\_size, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3)

#### Output shape:

2D tensor with shape (batch\_size, channels).

## \_\_init\_\_

\_\_init\_\_(  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.GlobalMaxPool1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D#top_of_page)
* [Class GlobalMaxPool1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D#class_globalmaxpool1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool1D#__init__)

## Class GlobalMaxPool1D

Global max pooling operation for temporal data.

### Aliases:

* Class tf.compat.v1.keras.layers.GlobalMaxPool1D
* Class tf.compat.v1.keras.layers.GlobalMaxPooling1D
* Class tf.compat.v2.keras.layers.GlobalMaxPool1D
* Class tf.compat.v2.keras.layers.GlobalMaxPooling1D
* Class tf.keras.layers.GlobalMaxPool1D
* Class tf.keras.layers.GlobalMaxPooling1D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

#### Arguments:

* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, steps, features) while channels\_first corresponds to inputs with shape (batch, features, steps).

#### Input shape:

* If data\_format='channels\_last': 3D tensor with shape: (batch\_size, steps, features)
* If data\_format='channels\_first': 3D tensor with shape: (batch\_size, features, steps)

#### Output shape:

2D tensor with shape (batch\_size, features).

## \_\_init\_\_

\_\_init\_\_(  
    data\_format='channels\_last',  
    \*\*kwargs  
)

# tf.keras.layers.GlobalMaxPool2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D#top_of_page)
* [Class GlobalMaxPool2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D#class_globalmaxpool2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GlobalMaxPool2D#__init__)

## Class GlobalMaxPool2D

Global max pooling operation for spatial data.

### Aliases:

* Class tf.compat.v1.keras.layers.GlobalMaxPool2D
* Class tf.compat.v1.keras.layers.GlobalMaxPooling2D
* Class tf.compat.v2.keras.layers.GlobalMaxPool2D
* Class tf.compat.v2.keras.layers.GlobalMaxPooling2D
* Class tf.keras.layers.GlobalMaxPool2D
* Class tf.keras.layers.GlobalMaxPooling2D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

#### Arguments:

* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 4D tensor with shape (batch\_size, rows, cols, channels).
* If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, rows, cols).

#### Output shape:

2D tensor with shape (batch\_size, channels).

## \_\_init\_\_

\_\_init\_\_(  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.GRU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRU#top_of_page)
* [Class GRU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRU#class_gru)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRU#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRU#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRU#__init__)

## Class GRU

Gated Recurrent Unit - Cho et al. 2014.

Inherits From: [GRU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/GRU)

### Aliases:

* Class tf.compat.v2.keras.layers.GRU
* Class tf.keras.layers.GRU

Defined in [python/keras/layers/recurrent\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent_v2.py).

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)

Based on available runtime hardware and constraints, this layer will choose different implementations (cuDNN-based or pure-TensorFlow) to maximize the performance. If a GPU is available and all the **Arguments** to the layer meet the requirement of the CuDNN kernel (see below for details), the layer will use a fast cuDNN implementation.

The requirements to use the cuDNN implementation are:

1. activation == 'tanh'
2. recurrent\_activation == 'sigmoid'
3. recurrent\_dropout == 0
4. unroll is False
5. use\_bias is True
6. reset\_after is True
7. Inputs are not masked or strictly right padded.

There are two variants of the GRU implementation. The default one is based on [v3](https://arxiv.org/abs/1406.1078v3) and has reset gate applied to hidden state before matrix multiplication. The other one is based on [original](https://arxiv.org/abs/1406.1078v1) and has the order reversed.

The second variant is compatible with CuDNNGRU (GPU-only) and allows inference on CPU. Thus it has separate biases for kernel and recurrent\_kernel. To use this variant, set 'reset\_after'=True and recurrent\_activation='sigmoid'.

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **recurrent\_activation**: Activation function to use for the recurrent step. Default: sigmoid (sigmoid). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.
* **implementation**: Implementation mode, either 1 or 2. Mode 1 will structure its operations as a larger number of smaller dot products and additions, whereas mode 2 will batch them into fewer, larger operations. These modes will have different performance profiles on different hardware and for different applications.
* **return\_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence.
* **return\_state**: Boolean. Whether to return the last state in addition to the output.
* **go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.
* **stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
* **unroll**: Boolean (default False). If True, the network will be unrolled, else a symbolic loop will be used. Unrolling can speed-up a RNN, although it tends to be more memory-intensive. Unrolling is only suitable for short sequences.
* **reset\_after**: GRU convention (whether to apply reset gate after or before matrix multiplication). False = "before", True = "after" (default and CuDNN compatible).

#### Call Arguments:

* **inputs**: A 3D tensor.
* **mask**: Binary tensor of shape (samples, timesteps) indicating whether a given timestep should be masked.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. This argument is passed to the cell when calling it. This is only relevant if dropout or recurrent\_dropout is used.
* **initial\_state**: List of initial state tensors to be passed to the first call of the cell.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    recurrent\_activation='sigmoid',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    implementation=2,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    unroll=False,  
    time\_major=False,  
    reset\_after=True,  
    \*\*kwargs  
)

## Properties

### activation

### bias\_constraint

### bias\_initializer

### bias\_regularizer

### dropout

### implementation

### kernel\_constraint

### kernel\_initializer

### kernel\_regularizer

### recurrent\_activation

### recurrent\_constraint

### recurrent\_dropout

### recurrent\_initializer

### recurrent\_regularizer

### reset\_after

### states

### units

### use\_bias

## Methods

### get\_dropout\_mask\_for\_cell

get\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the dropout mask for RNN cell's input.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### get\_initial\_state

get\_initial\_state(inputs)

### get\_recurrent\_dropout\_mask\_for\_cell

get\_recurrent\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the recurrent dropout mask for RNN cell.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### reset\_dropout\_mask

reset\_dropout\_mask()

Reset the cached dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_recurrent\_dropout\_mask

reset\_recurrent\_dropout\_mask()

Reset the cached recurrent dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_states

reset\_states(states=None)

# tf.keras.layers.GRUCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRUCell#top_of_page)
* [Class GRUCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRUCell#class_grucell)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRUCell#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRUCell#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/GRUCell#methods)

## Class GRUCell

Cell class for the GRU layer.

Inherits From: [GRUCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/GRUCell)

### Aliases:

* Class tf.compat.v2.keras.layers.GRUCell
* Class tf.keras.layers.GRUCell

Defined in [python/keras/layers/recurrent\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent_v2.py).

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **recurrent\_activation**: Activation function to use for the recurrent step. Default: sigmoid (sigmoid). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.
* **implementation**: Implementation mode, either 1 or 2. Mode 1 will structure its operations as a larger number of smaller dot products and additions, whereas mode 2 (default) will batch them into fewer, larger operations. These modes will have different performance profiles on different hardware and for different applications.
* **reset\_after**: GRU convention (whether to apply reset gate after or before matrix multiplication). False = "before", True = "after" (default and CuDNN compatible).

#### Call Arguments:

* **inputs**: A 2D tensor.
* **states**: List of state tensors corresponding to the previous timestep.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. Only relevant when dropout or recurrent\_dropout is used.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    recurrent\_activation='sigmoid',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    implementation=2,  
    reset\_after=True,  
    \*\*kwargs  
)

## Methods

### get\_dropout\_mask\_for\_cell

get\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the dropout mask for RNN cell's input.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

### get\_recurrent\_dropout\_mask\_for\_cell

get\_recurrent\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the recurrent dropout mask for RNN cell.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### reset\_dropout\_mask

reset\_dropout\_mask()

Reset the cached dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_recurrent\_dropout\_mask

reset\_recurrent\_dropout\_mask()

Reset the cached recurrent dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

# tf.keras.layers.InputLayer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputLayer#top_of_page)
* [Class InputLayer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputLayer#class_inputlayer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputLayer#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputLayer#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputLayer#__init__)

## Class InputLayer

Layer to be used as an entry point into a Network (a graph of layers).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.InputLayer
* Class tf.compat.v2.keras.layers.InputLayer
* Class tf.keras.layers.InputLayer

Defined in [python/keras/engine/input\_layer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/engine/input_layer.py).

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

It can either wrap an existing tensor (pass an input\_tensor argument) or create its a placeholder tensor (pass **Arguments** input\_shape, and optionally, dtype).

It is generally recommend to use the functional layer API via Input, (which creates an InputLayer) without directly using InputLayer.

#### Arguments:

* **input\_shape**: Shape tuple (not including the batch axis), or TensorShape instance (not including the batch axis).
* **batch\_size**: Optional input batch size (integer or None).
* **dtype**: Datatype of the input.
* **input\_tensor**: Optional tensor to use as layer input instead of creating a placeholder.
* **sparse**: Boolean, whether the placeholder created is meant to be sparse.
* **name**: Name of the layer (string).

## \_\_init\_\_

\_\_init\_\_(  
    input\_shape=None,  
    batch\_size=None,  
    dtype=None,  
    input\_tensor=None,  
    sparse=False,  
    name=None,  
    \*\*kwargs  
)

# tf.keras.layers.InputSpec

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#top_of_page)
* [Class InputSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#class_inputspec)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#methods)
  + [from\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#from_config)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/InputSpec#get_config)

## Class InputSpec

Specifies the ndim, dtype and shape of every input to a layer.

### Aliases:

* Class tf.compat.v1.keras.layers.InputSpec
* Class tf.compat.v1.layers.InputSpec
* Class tf.compat.v2.keras.layers.InputSpec
* Class tf.keras.layers.InputSpec

Defined in [python/keras/engine/input\_spec.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/engine/input_spec.py).

Every layer should expose (if appropriate) an input\_spec attribute: a list of instances of InputSpec (one per input tensor).

A None entry in a shape is compatible with any dimension, a None shape is compatible with any shape.

#### Arguments:

* **dtype**: Expected DataType of the input.
* **shape**: Shape tuple, expected shape of the input (may include None for unchecked axes).
* **ndim**: Integer, expected rank of the input.
* **max\_ndim**: Integer, maximum rank of the input.
* **min\_ndim**: Integer, minimum rank of the input.
* **axes**: Dictionary mapping integer axes to a specific dimension value.

## \_\_init\_\_

\_\_init\_\_(  
    dtype=None,  
    shape=None,  
    ndim=None,  
    max\_ndim=None,  
    min\_ndim=None,  
    axes=None  
)

## Methods

### from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

### get\_config

get\_config()

# tf.keras.layers.Lambda

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Lambda#top_of_page)
* [Class Lambda](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Lambda#class_lambda)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Lambda#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Lambda#__init__)

## Class Lambda

Wraps arbitrary expressions as a Layer object.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Lambda
* Class tf.compat.v2.keras.layers.Lambda
* Class tf.keras.layers.Lambda

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

The Lambda layer exists so that arbitrary TensorFlow functions can be used when constructing Sequential and Functional API models. Lambda layers are best suited for simple operations or quick experimentation. For more advanced use cases, subclassing keras.layers.Layer is preferred. One reason for this is that when saving a Model, Lambda layers are saved by serializing the Python bytecode, whereas subclassed Layers are saved via overriding their get\_config method and are thus more portable. Models that rely on subclassed Layers are also often easier to visualize and reason about.

#### Examples:

# add a x -> x^2 layer  
model.add(Lambda(lambda x: x \*\* 2))

# add a layer that returns the concatenation  
# of the positive part of the input and  
# the opposite of the negative part  
  
def antirectifier(x):  
    x -= K.mean(x, axis=1, keepdims=True)  
    x = K.l2\_normalize(x, axis=1)  
    pos = K.relu(x)  
    neg = K.relu(-x)  
    return K.concatenate([pos, neg], axis=1)  
  
model.add(Lambda(antirectifier))

Variables can be created within a Lambda layer. Like with other layers, these variables will be created only once and reused if the Lambda layer is called on new inputs. If creating more than one variable in a given Lambda instance, be sure to use a different name for each variable. Note that calling sublayers from within a Lambda is not supported.

Example of variable creation:

def linear\_transform(x):  
  v1 = tf.Variable(1., name='multiplier')  
  v2 = tf.Variable(0., name='bias')  
  return x\*v1 + v2  
  
linear\_layer = Lambda(linear\_transform)  
model.add(linear\_layer)  
model.add(keras.layers.Dense(10, activation='relu'))  
model.add(linear\_layer)  # Reuses existing Variables

Note that creating two instances of Lambda using the same function will not share Variables between the two instances. Each instance of Lambda will create and manage its own weights.

#### Arguments:

* **function**: The function to be evaluated. Takes input tensor as first argument.
* **output\_shape**: Expected output shape from function. This argument can be inferred if not explicitly provided. Can be a tuple or function. If a tuple, it only specifies the first dimension onward; sample dimension is assumed either the same as the input: output\_shape = (input\_shape[0], ) + output\_shape or, the input is None and the sample dimension is also None: output\_shape = (None, ) + output\_shape If a function, it specifies the entire shape as a function of the input shape: output\_shape = f(input\_shape)
* **mask**: Either None (indicating no masking) or a callable with the same signature as the compute\_mask layer method, or a tensor that will be returned as output mask regardless what the input is.
* **Arguments**: Optional dictionary of keyword **Arguments** to be passed to the function. Input shape: Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model. Output shape: Specified by output\_shape argument

## \_\_init\_\_

\_\_init\_\_(  
    function,  
    output\_shape=None,  
    mask=None,  
    **Arguments**=None,  
    \*\*kwargs  
)

# tf.keras.layers.Layer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer#top_of_page)
* [Class Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer#class_layer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer#properties)

## Class Layer

Base layer class.

Inherits From: [Module](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Module)

### Aliases:

* Class tf.compat.v1.keras.layers.Layer
* Class tf.compat.v2.keras.layers.Layer
* Class tf.keras.layers.Layer

Defined in [python/keras/engine/base\_layer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/engine/base_layer.py).

This is the class from which all layers inherit.

A layer is a class implementing common neural networks operations, such as convolution, batch norm, etc. These operations require managing weights, losses, updates, and inter-layer connectivity.

Users will just instantiate a layer and then treat it as a callable.

We recommend that descendants of Layer implement the following methods:

* \_\_init\_\_(): Save configuration in member variables
* build(): Called once from \_\_call\_\_, when we know the shapes of inputs and dtype. Should have the calls to add\_weight(), and then call the super's build() (which sets self.built = True, which is nice in case the user wants to call build() manually before the first \_\_call\_\_).
* call(): Called in \_\_call\_\_ after making sure build() has been called once. Should actually perform the logic of applying the layer to the input tensors (which should be passed in as the first argument).

#### Arguments:

* **trainable**: Boolean, whether the layer's variables should be trainable.
* **name**: String name of the layer.
* **dtype**: Default dtype of the layer's weights (default of None means use the type of the first input).
* **dynamic**: Set this to True if your layer should only be run eagerly, and should not be used to generate a static computation graph. This would be the case for a Tree-RNN or a recursive network, for example, or generally for any layer that manipulates tensors using Python control flow. If False, we assume that the layer can safely be used to generate a static computation graph.

Read-only properties: name: The name of the layer (string). dtype: Default dtype of the layer's weights (default of None means use the type of the first input). updates: List of update ops of this layer. losses: List of losses added by this layer. trainable\_weights: List of variables to be included in backprop. non\_trainable\_weights: List of variables that should not be included in backprop. weights: The concatenation of the lists trainable\_weights and non\_trainable\_weights (in this order).

#### Mutable properties:

* **trainable**: Whether the layer should be trained (boolean).
* **input\_spec**: Optional (list of) InputSpec object(s) specifying the constraints on inputs that can be accepted by the layer.

## \_\_init\_\_

\_\_init\_\_(  
    trainable=True,  
    name=None,  
    dtype=None,  
    dynamic=False,  
    \*\*kwargs  
)

## Properties

### activity\_regularizer

Optional regularizer function for the output of this layer.

### dtype

### dynamic

### input

Retrieves the input tensor(s) of a layer.

Only applicable if the layer has exactly one input, i.e. if it is connected to one incoming layer.

#### Returns:

Input tensor or list of input tensors.

#### Raises:

* **AttributeError**: if the layer is connected to more than one incoming layers.

#### Raises:

* **RuntimeError**: If called in Eager mode.
* **AttributeError**: If no inbound nodes are found.

### input\_mask

Retrieves the input mask tensor(s) of a layer.

Only applicable if the layer has exactly one inbound node, i.e. if it is connected to one incoming layer.

#### Returns:

Input mask tensor (potentially None) or list of input mask tensors.

#### Raises:

* **AttributeError**: if the layer is connected to more than one incoming layers.

### input\_shape

Retrieves the input shape(s) of a layer.

Only applicable if the layer has exactly one input, i.e. if it is connected to one incoming layer, or if all inputs have the same shape.

#### Returns:

Input shape, as an integer shape tuple (or list of shape tuples, one tuple per input tensor).

#### Raises:

* **AttributeError**: if the layer has no defined input\_shape.
* **RuntimeError**: if called in Eager mode.

### losses

Losses which are associated with this Layer.

Variable regularization tensors are created when this property is accessed, so it is eager safe: accessing losses under a [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) will propagate gradients back to the corresponding variables.

#### Returns:

A list of tensors.

### metrics

### name

### non\_trainable\_variables

### non\_trainable\_weights

### output

Retrieves the output tensor(s) of a layer.

Only applicable if the layer has exactly one output, i.e. if it is connected to one incoming layer.

#### Returns:

Output tensor or list of output tensors.

#### Raises:

* **AttributeError**: if the layer is connected to more than one incoming layers.
* **RuntimeError**: if called in Eager mode.

### output\_mask

Retrieves the output mask tensor(s) of a layer.

Only applicable if the layer has exactly one inbound node, i.e. if it is connected to one incoming layer.

#### Returns:

Output mask tensor (potentially None) or list of output mask tensors.

#### Raises:

* **AttributeError**: if the layer is connected to more than one incoming layers.

### output\_shape

Retrieves the output shape(s) of a layer.

Only applicable if the layer has one output, or if all outputs have the same shape.

#### Returns:

Output shape, as an integer shape tuple (or list of shape tuples, one tuple per output tensor).

#### Raises:

* **AttributeError**: if the layer has no defined output shape.
* **RuntimeError**: if called in Eager mode.

### trainable

### trainable\_variables

### trainable\_weights

### updates

### variables

Returns the list of all layer variables/weights.

Alias of self.weights.

#### Returns:

A list of variables.

### weights

Returns the list of all layer variables/weights.

#### Returns:

A list of variables.

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    inputs,  
    \*args,  
    \*\*kwargs  
)

Wraps call, applying pre- and post-processing steps.

#### Arguments:

* **inputs**: input tensor(s).
* **\*args**: additional positional **Arguments** to be passed to self.call.
* **\*\*kwargs**: additional keyword **Arguments** to be passed to self.call.

#### Returns:

Output tensor(s).

#### Note:

* The following optional keyword **Arguments** are reserved for specific uses:
  + training: Boolean scalar tensor of Python boolean indicating whether the call is meant for training or inference.
  + mask: Boolean input mask.
* If the layer's call method takes a mask argument (as some Keras layers do), its default value will be set to the mask generated for inputs by the previous layer (if input did come from a layer that generated a corresponding mask, i.e. if it came from a Keras layer with masking support.

#### Raises:

* **ValueError**: if the layer's call method returns None (an invalid value).

### add\_loss

add\_loss(  
    losses,  
    inputs=None  
)

Add loss tensor(s), potentially dependent on layer inputs.

Some losses (for instance, activity regularization losses) may be dependent on the inputs passed when calling a layer. Hence, when reusing the same layer on different inputs a and b, some entries in layer.losses may be dependent on a and some on b. This method automatically keeps track of dependencies.

This method can be used inside a subclassed layer or model's call function, in which case lossesshould be a Tensor or list of Tensors.

#### Example:

class MyLayer(tf.keras.layers.Layer):  
  def call(inputs, self):  
    self.add\_loss(tf.abs(tf.reduce\_mean(inputs)), inputs=True)  
    return inputs

This method can also be called directly on a Functional Model during construction. In this case, any loss Tensors passed to this Model must be symbolic and be able to be traced back to the model's Inputs. These losses become part of the model's topology and are tracked in get\_config.

#### Example:

inputs = tf.keras.Input(shape=(10,))  
x = tf.keras.layers.Dense(10)(inputs)  
outputs = tf.keras.layers.Dense(1)(x)  
model = tf.keras.Model(inputs, outputs)  
# Actvity regularization.  
model.add\_loss(tf.abs(tf.reduce\_mean(x)))

If this is not the case for your loss (if, for example, your loss references a Variable of one of the model's layers), you can wrap your loss in a zero-argument lambda. These losses are not tracked as part of the model's topology since they can't be serialized.

#### Example:

inputs = tf.keras.Input(shape=(10,))  
x = tf.keras.layers.Dense(10)(inputs)  
outputs = tf.keras.layers.Dense(1)(x)  
model = tf.keras.Model(inputs, outputs)  
# Weight regularization.  
model.add\_loss(lambda: tf.reduce\_mean(x.kernel))

The get\_losses\_for method allows to retrieve the losses relevant to a specific set of inputs.

#### Arguments:

* **losses**: Loss tensor, or list/tuple of tensors. Rather than tensors, losses may also be zero-argument callables which create a loss tensor.
* **inputs**: Ignored when executing eagerly. If anything other than None is passed, it signals the losses are conditional on some of the layer's inputs, and thus they should only be run where these inputs are available. This is the case for activity regularization losses, for instance. If Noneis passed, the losses are assumed to be unconditional, and will apply across all dataflows of the layer (e.g. weight regularization losses).

### add\_metric

add\_metric(  
    value,  
    aggregation=None,  
    name=None  
)

Adds metric tensor to the layer.

#### Args:

* **value**: Metric tensor.
* **aggregation**: Sample-wise metric reduction function. If aggregation=None, it indicates that the metric tensor provided has been aggregated already. eg, bin\_acc = BinaryAccuracy(name='acc') followed by model.add\_metric(bin\_acc(y\_true, y\_pred)). If aggregation='mean', the given metric tensor will be sample-wise reduced using mean function. eg, model.add\_metric(tf.reduce\_sum(outputs), name='output\_mean', aggregation='mean').
* **name**: String metric name.

#### Raises:

* **ValueError**: If aggregation is anything other than None or mean.

### add\_update

add\_update(  
    updates,  
    inputs=None  
)

Add update op(s), potentially dependent on layer inputs. (deprecated **Arguments**)

**Warning:** SOME **ARGUMENTS** ARE DEPRECATED: **(inputs)**. They will be removed in a future version. Instructions for updating: **inputs** is now automatically inferred

Weight updates (for instance, the updates of the moving mean and variance in a BatchNormalization layer) may be dependent on the inputs passed when calling a layer. Hence, when reusing the same layer on different inputs a and b, some entries in layer.updates may be dependent on a and some on b. This method automatically keeps track of dependencies.

The get\_updates\_for method allows to retrieve the updates relevant to a specific set of inputs.

This call is ignored when eager execution is enabled (in that case, variable updates are run on the fly and thus do not need to be tracked for later execution).

#### Arguments:

* **updates**: Update op, or list/tuple of update ops, or zero-arg callable that returns an update op. A zero-arg callable should be passed in order to disable running the updates by setting trainable=False on this Layer, when executing in Eager mode.
* **inputs**: Deprecated, will be automatically inferred.

### add\_variable

add\_variable(  
    \*args,  
    \*\*kwargs  
)

Alias for add\_weight.

### add\_weight

add\_weight(  
    name=None,  
    shape=None,  
    dtype=None,  
    initializer=None,  
    regularizer=None,  
    trainable=None,  
    constraint=None,  
    partitioner=None,  
    use\_resource=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE,  
    \*\*kwargs  
)

Adds a new variable to the layer.

#### Arguments:

* **name**: Variable name.
* **shape**: Variable shape. Defaults to scalar if unspecified.
* **dtype**: The type of the variable. Defaults to self.dtype or float32.
* **initializer**: initializer instance (callable).
* **regularizer**: regularizer instance (callable).
* **trainable**: whether the variable should be part of the layer's "trainable\_variables" (e.g. variables, biases) or "non\_trainable\_variables" (e.g. BatchNorm mean, stddev). Note, if the current variable scope is marked as non-trainable then this parameter is ignored and any added variables are also marked as non-trainable. trainable defaults to True unless synchronization is set to ON\_READ.
* **constraint**: constraint instance (callable).
* **partitioner**: Partitioner to be passed to the Trackable API.
* **use\_resource**: Whether to use ResourceVariable.
* **synchronization**: Indicates when a distributed a variable will be aggregated. Accepted values are constants defined in the class [tf.VariableSynchronization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/VariableSynchronization). By default the synchronization is set to AUTO and the current DistributionStrategy chooses when to synchronize. If synchronization is set to ON\_READ, trainable must not be set to True.
* **aggregation**: Indicates how a distributed variable will be aggregated. Accepted values are constants defined in the class [tf.VariableAggregation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/VariableAggregation).
* **\*\*kwargs**: Additional keyword **Arguments**. Accepted values are getter and collections.

#### Returns:

The created variable. Usually either a Variable or ResourceVariable instance. If partitioner is not None, a PartitionedVariable instance is returned.

#### Raises:

* **RuntimeError**: If called with partioned variable regularization and eager execution is enabled.
* **ValueError**: When giving unsupported dtype and no initializer or when trainable has been set to True with synchronization set as ON\_READ.

### apply

apply(  
    inputs,  
    \*args,  
    \*\*kwargs  
)

Apply the layer on a input.

This is an alias of self.\_\_call\_\_.

#### Arguments:

* **inputs**: Input tensor(s).
* **\*args**: additional positional **Arguments** to be passed to self.call.
* **\*\*kwargs**: additional keyword **Arguments** to be passed to self.call.

#### Returns:

Output tensor(s).

### build

build(input\_shape)

Creates the variables of the layer (optional, for subclass implementers).

This is a method that implementers of subclasses of Layer or Model can override if they need a state-creation step in-between layer instantiation and layer call.

This is typically used to create the weights of Layer subclasses.

#### Arguments:

* **input\_shape**: Instance of TensorShape, or list of instances of TensorShape if the layer expects a list of inputs (one instance per input).

### call

call(  
    inputs,  
    \*\*kwargs  
)

This is where the layer's logic lives.

#### Arguments:

* **inputs**: Input tensor, or list/tuple of input tensors.
* **\*\*kwargs**: Additional keyword **Arguments**.

#### Returns:

A tensor or list/tuple of tensors.

### compute\_mask

compute\_mask(  
    inputs,  
    mask=None  
)

Computes an output mask tensor.

#### Arguments:

* **inputs**: Tensor or list of tensors.
* **mask**: Tensor or list of tensors.

#### Returns:

None or a tensor (or list of tensors, one per output tensor of the layer).

### compute\_output\_shape

compute\_output\_shape(input\_shape)

Computes the output shape of the layer.

Assumes that the layer will be built to match that input shape provided.

#### Arguments:

* **input\_shape**: Shape tuple (tuple of integers) or list of shape tuples (one per output tensor of the layer). Shape tuples can include None for free dimensions, instead of an integer.

#### Returns:

An input shape tuple.

### compute\_output\_signature

compute\_output\_signature(input\_signature)

Compute the output tensor signature of the layer based on the inputs.

Unlike a TensorShape object, a TensorSpec object contains both shape and dtype information for a tensor. This method allows layers to provide output dtype information if it is different from the input dtype. For any layer that doesn't implement this function, the framework will fall back to use compute\_output\_shape, and will assume that the output dtype matches the input dtype.

#### Args:

* **input\_signature**: Single TensorSpec or nested structure of TensorSpec objects, describing a candidate input for the layer.

#### Returns:

Single TensorSpec or nested structure of TensorSpec objects, describing how the layer would transform the provided input.

### count\_params

count\_params()

Count the total number of scalars composing the weights.

#### Returns:

An integer count.

#### Raises:

* **ValueError**: if the layer isn't yet built (in which case its weights aren't yet defined).

### from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

Creates a layer from its config.

This method is the reverse of get\_config, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by set\_weights).

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.

#### Returns:

A layer instance.

### get\_config

get\_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by Network (one layer of abstraction above).

#### Returns:

Python dictionary.

### get\_input\_at

get\_input\_at(node\_index)

Retrieves the input tensor(s) of a layer at a given node.

#### Arguments:

* **node\_index**: Integer, index of the node from which to retrieve the attribute. E.g. node\_index=0will correspond to the first time the layer was called.

#### Returns:

A tensor (or list of tensors if the layer has multiple inputs).

#### Raises:

* **RuntimeError**: If called in Eager mode.

### get\_input\_mask\_at

get\_input\_mask\_at(node\_index)

Retrieves the input mask tensor(s) of a layer at a given node.

#### Arguments:

* **node\_index**: Integer, index of the node from which to retrieve the attribute. E.g. node\_index=0will correspond to the first time the layer was called.

#### Returns:

A mask tensor (or list of tensors if the layer has multiple inputs).

### get\_input\_shape\_at

get\_input\_shape\_at(node\_index)

Retrieves the input shape(s) of a layer at a given node.

#### Arguments:

* **node\_index**: Integer, index of the node from which to retrieve the attribute. E.g. node\_index=0will correspond to the first time the layer was called.

#### Returns:

A shape tuple (or list of shape tuples if the layer has multiple inputs).

#### Raises:

* **RuntimeError**: If called in Eager mode.

### get\_losses\_for

get\_losses\_for(inputs)

Retrieves losses relevant to a specific set of inputs.

#### Arguments:

* **inputs**: Input tensor or list/tuple of input tensors.

#### Returns:

List of loss tensors of the layer that depend on inputs.

### get\_output\_at

get\_output\_at(node\_index)

Retrieves the output tensor(s) of a layer at a given node.

#### Arguments:

* **node\_index**: Integer, index of the node from which to retrieve the attribute. E.g. node\_index=0will correspond to the first time the layer was called.

#### Returns:

A tensor (or list of tensors if the layer has multiple outputs).

#### Raises:

* **RuntimeError**: If called in Eager mode.

### get\_output\_mask\_at

get\_output\_mask\_at(node\_index)

Retrieves the output mask tensor(s) of a layer at a given node.

#### Arguments:

* **node\_index**: Integer, index of the node from which to retrieve the attribute. E.g. node\_index=0will correspond to the first time the layer was called.

#### Returns:

A mask tensor (or list of tensors if the layer has multiple outputs).

### get\_output\_shape\_at

get\_output\_shape\_at(node\_index)

Retrieves the output shape(s) of a layer at a given node.

#### Arguments:

* **node\_index**: Integer, index of the node from which to retrieve the attribute. E.g. node\_index=0will correspond to the first time the layer was called.

#### Returns:

A shape tuple (or list of shape tuples if the layer has multiple outputs).

#### Raises:

* **RuntimeError**: If called in Eager mode.

### get\_updates\_for

get\_updates\_for(inputs)

Retrieves updates relevant to a specific set of inputs.

#### Arguments:

* **inputs**: Input tensor or list/tuple of input tensors.

#### Returns:

List of update ops of the layer that depend on inputs.

### get\_weights

get\_weights()

Returns the current weights of the layer.

#### Returns:

Weights values as a list of numpy arrays.

### set\_weights

set\_weights(weights)

Sets the weights of the layer, from Numpy arrays.

#### Arguments:

* **weights**: a list of Numpy arrays. The number of arrays and their shape must match number of the dimensions of the weights of the layer (i.e. it should match the output of get\_weights).

#### Raises:

* **ValueError**: If the provided weights list does not match the layer's specifications.

# tf.keras.layers.LayerNormalization

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LayerNormalization#top_of_page)
* [Class LayerNormalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LayerNormalization#class_layernormalization)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LayerNormalization#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LayerNormalization#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LayerNormalization#__init__)

## Class LayerNormalization

Layer normalization layer (Ba et al., 2016).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.LayerNormalization
* Class tf.compat.v2.keras.layers.LayerNormalization
* Class tf.keras.layers.LayerNormalization

Defined in [python/keras/layers/normalization.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/normalization.py).

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

Normalize the activations of the previous layer for each given example in a batch independently, rather than across a batch like Batch Normalization. i.e. applies a transformation that maintains the mean activation within each example close to 0 and the activation standard deviation close to 1.

#### Arguments:

* **axis**: Integer or List/Tuple. The axis that should be normalized (typically the features axis).
* **epsilon**: Small float added to variance to avoid dividing by zero.
* **center**: If True, add offset of beta to normalized tensor. If False, beta is ignored.
* **scale**: If True, multiply by gamma. If False, gamma is not used. When the next layer is linear (also e.g. nn.relu), this can be disabled since the scaling will be done by the next layer.
* **beta\_initializer**: Initializer for the beta weight.
* **gamma\_initializer**: Initializer for the gamma weight.
* **beta\_regularizer**: Optional regularizer for the beta weight.
* **gamma\_regularizer**: Optional regularizer for the gamma weight.
* **beta\_constraint**: Optional constraint for the beta weight.
* **gamma\_constraint**: Optional constraint for the gamma weight.
* **trainable**: Boolean, if True the variables will be marked as trainable.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as input.

#### References:

* [Layer Normalization](https://arxiv.org/abs/1607.06450)

## \_\_init\_\_

\_\_init\_\_(  
    axis=-1,  
    epsilon=0.001,  
    center=True,  
    scale=True,  
    beta\_initializer='zeros',  
    gamma\_initializer='ones',  
    beta\_regularizer=None,  
    gamma\_regularizer=None,  
    beta\_constraint=None,  
    gamma\_constraint=None,  
    trainable=True,  
    name=None,  
    \*\*kwargs  
)

# tf.keras.layers.LeakyReLU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LeakyReLU#top_of_page)
* [Class LeakyReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LeakyReLU#class_leakyrelu)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LeakyReLU#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LeakyReLU#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LeakyReLU#__init__)

## Class LeakyReLU

Leaky version of a Rectified Linear Unit.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.LeakyReLU
* Class tf.compat.v2.keras.layers.LeakyReLU
* Class tf.keras.layers.LeakyReLU

Defined in [python/keras/layers/advanced\_activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/advanced_activations.py).

### Used in the tutorials:

* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

It allows a small gradient when the unit is not active: f(x) = alpha \* x for x < 0, f(x) = x for x >= 0.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as the input.

#### Arguments:

* **alpha**: Float >= 0. Negative slope coefficient.

## \_\_init\_\_

\_\_init\_\_(  
    alpha=0.3,  
    \*\*kwargs  
)

# tf.keras.layers.LocallyConnected1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected1D#top_of_page)
* [Class LocallyConnected1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected1D#class_locallyconnected1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected1D#__init__)

## Class LocallyConnected1D

Locally-connected layer for 1D inputs.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.LocallyConnected1D
* Class tf.compat.v2.keras.layers.LocallyConnected1D
* Class tf.keras.layers.LocallyConnected1D

Defined in [python/keras/layers/local.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/local.py).

The LocallyConnected1D layer works similarly to the Conv1D layer, except that weights are unshared, that is, a different set of filters is applied at each different patch of the input.

#### Example:

    # apply a unshared weight convolution 1d of length 3 to a sequence with  
    # 10 timesteps, with 64 output filters  
    model = Sequential()  
    model.add(LocallyConnected1D(64, 3, input\_shape=(10, 32)))  
    # now model.output\_shape == (None, 8, 64)  
    # add a new conv1d on top  
    model.add(LocallyConnected1D(32, 3))  
    # now model.output\_shape == (None, 6, 32)

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of a single integer, specifying the length of the 1D convolution window.
* **strides**: An integer or tuple/list of a single integer, specifying the stride length of the convolution. Specifying any stride value != 1 is incompatible with specifying any dilation\_ratevalue != 1.
* **padding**: Currently only supports "valid" (case-insensitive). "same" may be supported in the future.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, length, channels) while channels\_first corresponds to inputs with shape (batch, channels, length). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **implementation**: implementation mode, either 1 or 2. 1 loops over input spatial locations to perform the forward pass. It is memory-efficient but performs a lot of (small) ops.

2 stores layer weights in a dense but sparsely-populated 2D matrix and implements the forward pass as a single matrix-multiply. It uses a lot of RAM but performs few (large) ops.

Depending on the inputs, layer parameters, hardware, and tf.executing\_eagerly() one implementation can be dramatically faster (e.g. 50X) than another.

It is recommended to benchmark both in the setting of interest to pick the most efficient one (in terms of speed and memory usage).

Following scenarios could benefit from setting implementation=2: - eager execution; - inference; - running on CPU; - large amount of RAM available; - small models (few filters, small kernel); - using padding=same (only possible with implementation=2).

#### Input shape:

3D tensor with shape: (batch\_size, steps, input\_dim)

#### Output shape:

3D tensor with shape: (batch\_size, new\_steps, filters) steps value might have changed due to padding or strides.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=1,  
    padding='valid',  
    data\_format=None,  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    implementation=1,  
    \*\*kwargs  
)

# tf.keras.layers.LocallyConnected2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected2D#top_of_page)
* [Class LocallyConnected2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected2D#class_locallyconnected2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LocallyConnected2D#__init__)

## Class LocallyConnected2D

Locally-connected layer for 2D inputs.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.LocallyConnected2D
* Class tf.compat.v2.keras.layers.LocallyConnected2D
* Class tf.keras.layers.LocallyConnected2D

Defined in [python/keras/layers/local.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/local.py).

The LocallyConnected2D layer works similarly to the Conv2D layer, except that weights are unshared, that is, a different set of filters is applied at each different patch of the input.

#### Examples:

    # apply a 3x3 unshared weights convolution with 64 output filters on a  
    32x32 image  
    # with `data\_format="channels\_last"`:  
    model = Sequential()  
    model.add(LocallyConnected2D(64, (3, 3), input\_shape=(32, 32, 3)))  
    # now model.output\_shape == (None, 30, 30, 64)  
    # notice that this layer will consume (30\*30)\*(3\*3\*3\*64) + (30\*30)\*64  
    parameters  
  
    # add a 3x3 unshared weights convolution on top, with 32 output filters:  
    model.add(LocallyConnected2D(32, (3, 3)))  
    # now model.output\_shape == (None, 28, 28, 32)

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 2 integers, specifying the width and height of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the width and height. Can be a single integer to specify the same value for all spatial dimensions.
* **padding**: Currently only support "valid" (case-insensitive). "same" will be supported in future.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").
* **kernel\_constraint**: Constraint function applied to the kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **implementation**: implementation mode, either 1 or 2. 1 loops over input spatial locations to perform the forward pass. It is memory-efficient but performs a lot of (small) ops.

2 stores layer weights in a dense but sparsely-populated 2D matrix and implements the forward pass as a single matrix-multiply. It uses a lot of RAM but performs few (large) ops.

Depending on the inputs, layer parameters, hardware, and tf.executing\_eagerly() one implementation can be dramatically faster (e.g. 50X) than another.

It is recommended to benchmark both in the setting of interest to pick the most efficient one (in terms of speed and memory usage).

Following scenarios could benefit from setting implementation=2: - eager execution; - inference; - running on CPU; - large amount of RAM available; - small models (few filters, small kernel); - using padding=same (only possible with implementation=2).

#### Input shape:

4D tensor with shape: (samples, channels, rows, cols) if data\_format='channels\_first' or 4D tensor with shape: (samples, rows, cols, channels) if data\_format='channels\_last'.

#### Output shape:

4D tensor with shape: (samples, filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4D tensor with shape: (samples, new\_rows, new\_cols, filters) if data\_format='channels\_last'. rows and cols values might have changed due to padding.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    activation=None,  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    bias\_constraint=None,  
    implementation=1,  
    \*\*kwargs  
)

# tf.keras.layers.LSTM

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM#top_of_page)
* [Class LSTM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM#class_lstm)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM#used_in_the_tutorials)

## Class LSTM

Long Short-Term Memory layer - Hochreiter 1997.

Inherits From: [LSTM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/LSTM)

### Aliases:

* Class tf.compat.v2.keras.layers.LSTM
* Class tf.keras.layers.LSTM

Defined in [python/keras/layers/recurrent\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent_v2.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Load text with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/text)
* [Text classification with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_classification_rnn)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)

Based on available runtime hardware and constraints, this layer will choose different implementations (cuDNN-based or pure-TensorFlow) to maximize the performance. If a GPU is available and all the **Arguments** to the layer meet the requirement of the CuDNN kernel (see below for details), the layer will use a fast cuDNN implementation.

The requirements to use the cuDNN implementation are:

1. activation == 'tanh'
2. recurrent\_activation == 'sigmoid'
3. recurrent\_dropout == 0
4. unroll is False
5. use\_bias is True
6. Inputs are not masked or strictly right padded.

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **recurrent\_activation**: Activation function to use for the recurrent step. Default: sigmoid (sigmoid). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs..
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state..
* **bias\_initializer**: Initializer for the bias vector.
* **unit\_forget\_bias**: Boolean. If True, add 1 to the bias of the forget gate at initialization. Setting it to true will also force bias\_initializer="zeros". This is recommended in [Jozefowicz et al.](http://www.jmlr.org/proceedings/papers/v37/jozefowicz15.pdf).
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.
* **implementation**: Implementation mode, either 1 or 2. Mode 1 will structure its operations as a larger number of smaller dot products and additions, whereas mode 2 will batch them into fewer, larger operations. These modes will have different performance profiles on different hardware and for different applications.
* **return\_sequences**: Boolean. Whether to return the last output. in the output sequence, or the full sequence.
* **return\_state**: Boolean. Whether to return the last state in addition to the output.
* **go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.
* **stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
* **unroll**: Boolean (default False). If True, the network will be unrolled, else a symbolic loop will be used. Unrolling can speed-up a RNN, although it tends to be more memory-intensive. Unrolling is only suitable for short sequences.

#### Call Arguments:

* **inputs**: A 3D tensor.
* **mask**: Binary tensor of shape (samples, timesteps) indicating whether a given timestep should be masked.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. This argument is passed to the cell when calling it. This is only relevant if dropout or recurrent\_dropout is used.
* **initial\_state**: List of initial state tensors to be passed to the first call of the cell.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    recurrent\_activation='sigmoid',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    unit\_forget\_bias=True,  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    implementation=2,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    time\_major=False,  
    unroll=False,  
    \*\*kwargs  
)

## Properties

### activation

### bias\_constraint

### bias\_initializer

### bias\_regularizer

### dropout

### implementation

### kernel\_constraint

### kernel\_initializer

### kernel\_regularizer

### recurrent\_activation

### recurrent\_constraint

### recurrent\_dropout

### recurrent\_initializer

### recurrent\_regularizer

### states

### unit\_forget\_bias

### units

### use\_bias

## Methods

### get\_dropout\_mask\_for\_cell

get\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the dropout mask for RNN cell's input.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### get\_initial\_state

get\_initial\_state(inputs)

### get\_recurrent\_dropout\_mask\_for\_cell

get\_recurrent\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the recurrent dropout mask for RNN cell.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### reset\_dropout\_mask

reset\_dropout\_mask()

Reset the cached dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_recurrent\_dropout\_mask

reset\_recurrent\_dropout\_mask()

Reset the cached recurrent dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_states

reset\_states(states=None)

# tf.keras.layers.LSTMCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTMCell#top_of_page)
* [Class LSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTMCell#class_lstmcell)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTMCell#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTMCell#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTMCell#__init__)

## Class LSTMCell

Cell class for the LSTM layer.

Inherits From: [LSTMCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/LSTMCell)

### Aliases:

* Class tf.compat.v2.keras.layers.LSTMCell
* Class tf.keras.layers.LSTMCell

Defined in [python/keras/layers/recurrent\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent_v2.py).

### Used in the guide:

* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **recurrent\_activation**: Activation function to use for the recurrent step. Default: sigmoid (sigmoid). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.
* **bias\_initializer**: Initializer for the bias vector.
* **unit\_forget\_bias**: Boolean. If True, add 1 to the bias of the forget gate at initialization. Setting it to true will also force bias\_initializer="zeros". This is recommended in [Jozefowicz et al.](http://www.jmlr.org/proceedings/papers/v37/jozefowicz15.pdf)
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.
* **implementation**: Implementation mode, either 1 or 2. Mode 1 will structure its operations as a larger number of smaller dot products and additions, whereas mode 2 (default) will batch them into fewer, larger operations. These modes will have different performance profiles on different hardware and for different applications.

#### Call Arguments:

* **inputs**: A 2D tensor.
* **states**: List of state tensors corresponding to the previous timestep.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. Only relevant when dropout or recurrent\_dropout is used.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    recurrent\_activation='sigmoid',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    unit\_forget\_bias=True,  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    implementation=2,  
    \*\*kwargs  
)

## Methods

### get\_dropout\_mask\_for\_cell

get\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the dropout mask for RNN cell's input.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

### get\_recurrent\_dropout\_mask\_for\_cell

get\_recurrent\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the recurrent dropout mask for RNN cell.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### reset\_dropout\_mask

reset\_dropout\_mask()

Reset the cached dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_recurrent\_dropout\_mask

reset\_recurrent\_dropout\_mask()

Reset the cached recurrent dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

# tf.keras.layers.Masking

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Masking#top_of_page)
* [Class Masking](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Masking#class_masking)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Masking#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Masking#__init__)

## Class Masking

Masks a sequence by using a mask value to skip timesteps.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Masking
* Class tf.compat.v2.keras.layers.Masking
* Class tf.keras.layers.Masking

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

For each timestep in the input tensor (dimension #1 in the tensor), if all values in the input tensor at that timestep are equal to mask\_value, then the timestep will be masked (skipped) in all downstream layers (as long as they support masking).

If any downstream layer does not support masking yet receives such an input mask, an exception will be raised.

#### Example:

Consider a Numpy data array x of shape (samples, timesteps, features), to be fed to an LSTM layer. You want to mask timestep #3 and #5 because you lack data for these timesteps. You can:

* Set x[:, 3, :] = 0. and x[:, 5, :] = 0.
* Insert a Masking layer with mask\_value=0. before the LSTM layer:

model = Sequential()  
model.add(Masking(mask\_value=0., input\_shape=(timesteps, features)))  
model.add(LSTM(32))

## \_\_init\_\_

\_\_init\_\_(  
    mask\_value=0.0,  
    \*\*kwargs  
)

tf.keras.layers.Maximum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Maximum#top_of_page)
* [Class Maximum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Maximum#class_maximum)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Maximum#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Maximum#__init__)

Class Maximum

Layer that computes the maximum (element-wise) a list of inputs.

Aliases:

* Class tf.compat.v1.keras.layers.Maximum
* Class tf.compat.v2.keras.layers.Maximum
* Class tf.keras.layers.Maximum

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

It takes as input a list of tensors, all of the same shape, and returns a single tensor (also of the same shape).

\_\_init\_\_

\_\_init\_\_(\*\*kwargs)

# tf.keras.layers.maximum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/maximum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/maximum#aliases)

Functional interface to the Maximum layer.

### Aliases:

* tf.compat.v1.keras.layers.maximum
* tf.compat.v2.keras.layers.maximum
* tf.keras.layers.maximum

tf.keras.layers.maximum(  
    inputs,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the element-wise maximum of the inputs.

# tf.keras.layers.MaxPool1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool1D#top_of_page)
* [Class MaxPool1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool1D#class_maxpool1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool1D#__init__)

## Class MaxPool1D

Max pooling operation for temporal data.

### Aliases:

* Class tf.compat.v1.keras.layers.MaxPool1D
* Class tf.compat.v1.keras.layers.MaxPooling1D
* Class tf.compat.v2.keras.layers.MaxPool1D
* Class tf.compat.v2.keras.layers.MaxPooling1D
* Class tf.keras.layers.MaxPool1D
* Class tf.keras.layers.MaxPooling1D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

#### Arguments:

* **pool\_size**: Integer, size of the max pooling windows.
* **strides**: Integer, or None. Factor by which to downscale. E.g. 2 will halve the input. If None, it will default to pool\_size.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, steps, features) while channels\_first corresponds to inputs with shape (batch, features, steps).

#### Input shape:

* If data\_format='channels\_last': 3D tensor with shape (batch\_size, steps, features).
* If data\_format='channels\_first': 3D tensor with shape (batch\_size, features, steps).

#### Output shape:

* If data\_format='channels\_last': 3D tensor with shape (batch\_size, downsampled\_steps, features).
* If data\_format='channels\_first': 3D tensor with shape (batch\_size, features, downsampled\_steps).

## \_\_init\_\_

\_\_init\_\_(  
    pool\_size=2,  
    strides=None,  
    padding='valid',  
    data\_format='channels\_last',  
    \*\*kwargs  
)

# tf.keras.layers.MaxPool2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D#top_of_page)
* [Class MaxPool2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D#class_maxpool2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool2D#__init__)

## Class MaxPool2D

Max pooling operation for spatial data.

### Aliases:

* Class tf.compat.v1.keras.layers.MaxPool2D
* Class tf.compat.v1.keras.layers.MaxPooling2D
* Class tf.compat.v2.keras.layers.MaxPool2D
* Class tf.compat.v2.keras.layers.MaxPooling2D
* Class tf.keras.layers.MaxPool2D
* Class tf.keras.layers.MaxPooling2D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Convolutional Neural Networks](https://www.tensorflow.org/beta/tutorials/images/intro_to_cnns)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Multi-worker Training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_keras)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

#### Arguments:

* **pool\_size**: integer or tuple of 2 integers, factors by which to downscale (vertical, horizontal).(2, 2) will halve the input in both spatial dimension. If only one integer is specified, the same window length will be used for both dimensions.
* **strides**: Integer, tuple of 2 integers, or None. Strides values. If None, it will default to pool\_size.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 4D tensor with shape (batch\_size, rows, cols, channels).
* If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, rows, cols).

#### Output shape:

* If data\_format='channels\_last': 4D tensor with shape (batch\_size, pooled\_rows, pooled\_cols, channels).
* If data\_format='channels\_first': 4D tensor with shape (batch\_size, channels, pooled\_rows, pooled\_cols).

## \_\_init\_\_

\_\_init\_\_(  
    pool\_size=(2, 2),  
    strides=None,  
    padding='valid',  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.MaxPool3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool3D#top_of_page)
* [Class MaxPool3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool3D#class_maxpool3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/MaxPool3D#__init__)

## Class MaxPool3D

Max pooling operation for 3D data (spatial or spatio-temporal).

### Aliases:

* Class tf.compat.v1.keras.layers.MaxPool3D
* Class tf.compat.v1.keras.layers.MaxPooling3D
* Class tf.compat.v2.keras.layers.MaxPool3D
* Class tf.compat.v2.keras.layers.MaxPooling3D
* Class tf.keras.layers.MaxPool3D
* Class tf.keras.layers.MaxPooling3D

Defined in [python/keras/layers/pooling.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/pooling.py).

#### Arguments:

* **pool\_size**: Tuple of 3 integers, factors by which to downscale (dim1, dim2, dim3). (2, 2, 2)will halve the size of the 3D input in each dimension.
* **strides**: tuple of 3 integers, or None. Strides values.
* **padding**: One of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

* If data\_format='channels\_last': 5D tensor with shape: (batch\_size, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels)
* If data\_format='channels\_first': 5D tensor with shape: (batch\_size, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3)

#### Output shape:

* If data\_format='channels\_last': 5D tensor with shape: (batch\_size, pooled\_dim1, pooled\_dim2, pooled\_dim3, channels)
* If data\_format='channels\_first': 5D tensor with shape: (batch\_size, channels, pooled\_dim1, pooled\_dim2, pooled\_dim3)

## \_\_init\_\_

\_\_init\_\_(  
    pool\_size=(2, 2, 2),  
    strides=None,  
    padding='valid',  
    data\_format=None,  
    \*\*kwargs  
)

tf.keras.layers.Minimum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Minimum#top_of_page)
* [Class Minimum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Minimum#class_minimum)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Minimum#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Minimum#__init__)

Class Minimum

Layer that computes the minimum (element-wise) a list of inputs.

Aliases:

* Class tf.compat.v1.keras.layers.Minimum
* Class tf.compat.v2.keras.layers.Minimum
* Class tf.keras.layers.Minimum

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

It takes as input a list of tensors, all of the same shape, and returns a single tensor (also of the same shape).

\_\_init\_\_

\_\_init\_\_(\*\*kwargs)

# tf.keras.layers.minimum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/minimum#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/minimum#aliases)

Functional interface to the Minimum layer.

### Aliases:

* tf.compat.v1.keras.layers.minimum
* tf.compat.v2.keras.layers.minimum
* tf.keras.layers.minimum

tf.keras.layers.minimum(  
    inputs,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the element-wise minimum of the inputs.

tf.keras.layers.Multiply

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Multiply#top_of_page)
* [Class Multiply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Multiply#class_multiply)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Multiply#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Multiply#__init__)

Class Multiply

Layer that multiplies (element-wise) a list of inputs.

Aliases:

* Class tf.compat.v1.keras.layers.Multiply
* Class tf.compat.v2.keras.layers.Multiply
* Class tf.keras.layers.Multiply

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

It takes as input a list of tensors, all of the same shape, and returns a single tensor (also of the same shape).

\_\_init\_\_

\_\_init\_\_(\*\*kwargs)

# tf.keras.layers.multiply

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/multiply#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/multiply#aliases)

Functional interface to the Multiply layer.

### Aliases:

* tf.compat.v1.keras.layers.multiply
* tf.compat.v2.keras.layers.multiply
* tf.keras.layers.multiply

tf.keras.layers.multiply(  
    inputs,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

#### Arguments:

* **inputs**: A list of input tensors (at least 2).
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the element-wise product of the inputs.

# tf.keras.layers.Permute

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Permute#top_of_page)
* [Class Permute](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Permute#class_permute)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Permute#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Permute#__init__)

## Class Permute

Permutes the dimensions of the input according to a given pattern.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Permute
* Class tf.compat.v2.keras.layers.Permute
* Class tf.keras.layers.Permute

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

Useful for e.g. connecting RNNs and convnets together.

#### Example:

model = Sequential()  
model.add(Permute((2, 1), input\_shape=(10, 64)))  
# now: model.output\_shape == (None, 64, 10)  
# note: `None` is the batch dimension

#### Arguments:

* **dims**: Tuple of integers. Permutation pattern, does not include the samples dimension. Indexing starts at 1. For instance, (2, 1) permutes the first and second dimensions of the input.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same as the input shape, but with the dimensions re-ordered according to the specified pattern.

## \_\_init\_\_

\_\_init\_\_(  
    dims,  
    \*\*kwargs  
)

# tf.keras.layers.PReLU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/PReLU#top_of_page)
* [Class PReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/PReLU#class_prelu)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/PReLU#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/PReLU#__init__)

## Class PReLU

Parametric Rectified Linear Unit.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.PReLU
* Class tf.compat.v2.keras.layers.PReLU
* Class tf.keras.layers.PReLU

Defined in [python/keras/layers/advanced\_activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/advanced_activations.py).

#### It follows:

f(x) = alpha \* x for x < 0, f(x) = x for x >= 0, where alpha is a learned array with the same shape as x.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as the input.

#### Arguments:

* **alpha\_initializer**: Initializer function for the weights.
* **alpha\_regularizer**: Regularizer for the weights.
* **alpha\_constraint**: Constraint for the weights.
* **shared\_axes**: The axes along which to share learnable parameters for the activation function. For example, if the incoming feature maps are from a 2D convolution with output shape (batch, height, width, channels), and you wish to share parameters across space so that each filter only has one set of parameters, set shared\_axes=[1, 2].

## \_\_init\_\_

\_\_init\_\_(  
    alpha\_initializer='zeros',  
    alpha\_regularizer=None,  
    alpha\_constraint=None,  
    shared\_axes=None,  
    \*\*kwargs  
)

# tf.keras.layers.ReLU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ReLU#top_of_page)
* [Class ReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ReLU#class_relu)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ReLU#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ReLU#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ReLU#__init__)

## Class ReLU

Rectified Linear Unit activation function.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ReLU
* Class tf.compat.v2.keras.layers.ReLU
* Class tf.keras.layers.ReLU

Defined in [python/keras/layers/advanced\_activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/advanced_activations.py).

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

With default values, it returns element-wise max(x, 0).

Otherwise, it follows: f(x) = max\_value for x >= max\_value, f(x) = x for threshold <= x < max\_value, f(x) = negative\_slope \* (x - threshold) otherwise.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as the input.

#### Arguments:

* **max\_value**: Float >= 0. Maximum activation value.
* **negative\_slope**: Float >= 0. Negative slope coefficient.
* **threshold**: Float. Threshold value for thresholded activation.

## \_\_init\_\_

\_\_init\_\_(  
    max\_value=None,  
    negative\_slope=0,  
    threshold=0,  
    \*\*kwargs  
)

# tf.keras.layers.RepeatVector

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RepeatVector#top_of_page)
* [Class RepeatVector](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RepeatVector#class_repeatvector)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RepeatVector#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RepeatVector#__init__)

## Class RepeatVector

Repeats the input n times.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.RepeatVector
* Class tf.compat.v2.keras.layers.RepeatVector
* Class tf.keras.layers.RepeatVector

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

#### Example:

model = Sequential()  
model.add(Dense(32, input\_dim=32))  
# now: model.output\_shape == (None, 32)  
# note: `None` is the batch dimension  
  
model.add(RepeatVector(3))  
# now: model.output\_shape == (None, 3, 32)

#### Arguments:

* **n**: Integer, repetition factor.

#### Input shape:

2D tensor of shape (num\_samples, features).

#### Output shape:

3D tensor of shape (num\_samples, n, features).

## \_\_init\_\_

\_\_init\_\_(  
    n,  
    \*\*kwargs  
)

# tf.keras.layers.Reshape

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape#top_of_page)
* [Class Reshape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape#class_reshape)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Reshape#__init__)

## Class Reshape

Reshapes an output to a certain shape.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Reshape
* Class tf.compat.v2.keras.layers.Reshape
* Class tf.keras.layers.Reshape

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)

#### Arguments:

* **target\_shape**: Target shape. Tuple of integers, does not include the samples dimension (batch size).

#### Input shape:

Arbitrary, although all dimensions in the input shaped must be fixed. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

(batch\_size,) + target\_shape

#### Example:

# as first layer in a Sequential model  
model = Sequential()  
model.add(Reshape((3, 4), input\_shape=(12,)))  
# now: model.output\_shape == (None, 3, 4)  
# note: `None` is the batch dimension  
  
# as intermediate layer in a Sequential model  
model.add(Reshape((6, 2)))  
# now: model.output\_shape == (None, 6, 2)  
  
# also supports shape inference using `-1` as dimension  
model.add(Reshape((-1, 2, 2)))  
# now: model.output\_shape == (None, 3, 2, 2)

## \_\_init\_\_

\_\_init\_\_(  
    target\_shape,  
    \*\*kwargs  
)

# tf.keras.layers.RNN

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN#top_of_page)
* [Class RNN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN#class_rnn)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN#properties)

## Class RNN

Base class for recurrent layers.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.RNN
* Class tf.compat.v2.keras.layers.RNN
* Class tf.keras.layers.RNN

Defined in [python/keras/layers/recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent.py).

#### Arguments:

* **cell**: A RNN cell instance or a list of RNN cell instances. A RNN cell is a class that has:
  + A call(input\_at\_t, states\_at\_t) method, returning (output\_at\_t, states\_at\_t\_plus\_1). The call method of the cell can also take the optional argument constants, see section "Note on passing external constants" below.
  + A state\_size attribute. This can be a single integer (single state) in which case it is the size of the recurrent state. This can also be a list/tuple of integers (one size per state). The state\_size can also be TensorShape or tuple/list of TensorShape, to represent high dimension state.
  + A output\_size attribute. This can be a single integer or a TensorShape, which represent the shape of the output. For backward compatible reason, if this attribute is not available for the cell, the value will be inferred by the first element of the state\_size.
  + A get\_initial\_state(inputs=None, batch\_size=None, dtype=None) method that creates a tensor meant to be fed to call() as the initial state, if the user didn't specify any initial state via other means. The returned initial state should have a shape of [batch\_size, cell.state\_size]. The cell might choose to create a tensor full of zeros, or full of other values based on the cell's implementation. inputs is the input tensor to the RNN layer, which should contain the batch size as its shape[0], and also dtype. Note that the shape[0] might be None during the graph construction. Either the inputs or the pair of batch\_size and dtype are provided. batch\_size is a scalar tensor that represents the batch size of the inputs. dtype is [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) that represents the dtype of the inputs. For backward compatible reason, if this method is not implemented by the cell, the RNN layer will create a zero filled tensor with the size of [batch\_size, cell.state\_size]. In the case that cell is a list of RNN cell instances, the cells will be stacked on top of each other in the RNN, resulting in an efficient stacked RNN.
* **return\_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence.
* **return\_state**: Boolean. Whether to return the last state in addition to the output.
* **go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.
* **stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
* **unroll**: Boolean (default False). If True, the network will be unrolled, else a symbolic loop will be used. Unrolling can speed-up a RNN, although it tends to be more memory-intensive. Unrolling is only suitable for short sequences.
* **time\_major**: The shape format of the inputs and outputs tensors. If True, the inputs and outputs will be in shape (timesteps, batch, ...), whereas in the False case, it will be(batch, timesteps, ...). Using time\_major = True is a bit more efficient because it avoids transposes at the beginning and end of the RNN calculation. However, most TensorFlow data is batch-major, so by default this function accepts input and emits output in batch-major form.

#### Call Arguments:

* **inputs**: Input tensor.
* **mask**: Binary tensor of shape (samples, timesteps) indicating whether a given timestep should be masked.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. This argument is passed to the cell when calling it. This is for use with cells that use dropout.
* **initial\_state**: List of initial state tensors to be passed to the first call of the cell.
* **constants**: List of constant tensors to be passed to the cell at each timestep.

#### Input shape:

N-D tensor with shape (batch\_size, timesteps, ...) or (timesteps, batch\_size, ...) when time\_major is True.

#### Output shape:

* If return\_state: a list of tensors. The first tensor is the output. The remaining tensors are the last states, each with shape (batch\_size, state\_size), where state\_size could be a high dimension tensor shape.
* If return\_sequences: N-D tensor with shape (batch\_size, timesteps, output\_size), where output\_size could be a high dimension tensor shape, or (timesteps, batch\_size, output\_size) when time\_major is True.
* Else, N-D tensor with shape (batch\_size, output\_size), where output\_size could be a high dimension tensor shape.

#### Masking:

This layer supports masking for input data with a variable number of timesteps. To introduce masks to your data, use an [Embedding](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/embeddings) layer with the mask\_zero parameter set to True.

Note on using statefulness in RNNs: You can set RNN layers to be 'stateful', which means that the states computed for the samples in one batch will be reused as initial states for the samples in the next batch. This assumes a one-to-one mapping between samples in different successive batches.

To enable statefulness: - Specify stateful=True in the layer constructor. - Specify a fixed batch size for your model, by passing If sequential model: batch\_input\_shape=(...) to the first layer in your model. Else for functional model with 1 or more Input layers: batch\_shape=(...) to all the first layers in your model. This is the expected shape of your inputs including the batch size. It should be a tuple of integers, e.g. (32, 10, 100). - Specify shuffle=False when calling fit().

To reset the states of your model, call .reset\_states() on either a specific layer, or on your entire model.

Note on specifying the initial state of RNNs: You can specify the initial state of RNN layers symbolically by calling them with the keyword argument initial\_state. The value of initial\_state should be a tensor or list of tensors representing the initial state of the RNN layer.

You can specify the initial state of RNN layers numerically by calling reset\_states with the keyword argument states. The value of states should be a numpy array or list of numpy arrays representing the initial state of the RNN layer.

Note on passing external constants to RNNs: You can pass "external" constants to the cell using the constants keyword argument of RNN.\_\_call\_\_ (as well as RNN.call) method. This requires that the cell.call method accepts the same keyword argument constants. Such constants can be used to condition the cell transformation on additional static inputs (not changing over time), a.k.a. an attention mechanism.

#### Examples:

# First, let's define a RNN Cell, as a layer subclass.  
  
class MinimalRNNCell(keras.layers.Layer):  
  
    def \_\_init\_\_(self, units, \*\*kwargs):  
        self.units = units  
        self.state\_size = units  
        super(MinimalRNNCell, self).\_\_init\_\_(\*\*kwargs)  
  
    def build(self, input\_shape):  
        self.kernel = self.add\_weight(shape=(input\_shape[-1], self.units),  
                                      initializer='uniform',  
                                      name='kernel')  
        self.recurrent\_kernel = self.add\_weight(  
            shape=(self.units, self.units),  
            initializer='uniform',  
            name='recurrent\_kernel')  
        self.built = True  
  
    def call(self, inputs, states):  
        prev\_output = states[0]  
        h = K.dot(inputs, self.kernel)  
        output = h + K.dot(prev\_output, self.recurrent\_kernel)  
        return output, [output]  
  
# Let's use this cell in a RNN layer:  
  
cell = MinimalRNNCell(32)  
x = keras.Input((None, 5))  
layer = RNN(cell)  
y = layer(x)  
  
# Here's how to use the cell to build a stacked RNN:  
  
cells = [MinimalRNNCell(32), MinimalRNNCell(64)]  
x = keras.Input((None, 5))  
layer = RNN(cells)  
y = layer(x)

## \_\_init\_\_

\_\_init\_\_(  
    cell,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    unroll=False,  
    time\_major=False,  
    \*\*kwargs  
)

## Properties

### states

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states

reset\_states(states=None)

# tf.keras.layers.SeparableConv1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv1D#top_of_page)
* [Class SeparableConv1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv1D#class_separableconv1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv1D#__init__)

## Class SeparableConv1D

Depthwise separable 1D convolution.

### Aliases:

* Class tf.compat.v1.keras.layers.SeparableConv1D
* Class tf.compat.v1.keras.layers.SeparableConvolution1D
* Class tf.compat.v2.keras.layers.SeparableConv1D
* Class tf.compat.v2.keras.layers.SeparableConvolution1D
* Class tf.keras.layers.SeparableConv1D
* Class tf.keras.layers.SeparableConvolution1D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

This layer performs a depthwise convolution that acts separately on channels, followed by a pointwise convolution that mixes channels. If use\_bias is True and a bias initializer is provided, it adds a bias vector to the output. It then optionally applies an activation function to produce the final output.

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of filters in the convolution).
* **kernel\_size**: A single integer specifying the spatial dimensions of the filters.
* **strides**: A single integer specifying the strides of the convolution. Specifying any stridevalue != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: One of "valid", "same", or "causal" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, length, channels) while channels\_first corresponds to inputs with shape (batch, channels, length).
* **dilation\_rate**: A single integer, specifying the dilation rate to use for dilated convolution. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any stride value != 1.
* **depth\_multiplier**: The number of depthwise convolution output channels for each input channel. The total number of depthwise convolution output channels will be equal to num\_filters\_in \* depth\_multiplier.
* **activation**: Activation function. Set it to None to maintain a linear activation.
* **use\_bias**: Boolean, whether the layer uses a bias.
* **depthwise\_initializer**: An initializer for the depthwise convolution kernel.
* **pointwise\_initializer**: An initializer for the pointwise convolution kernel.
* **bias\_initializer**: An initializer for the bias vector. If None, the default initializer will be used.
* **depthwise\_regularizer**: Optional regularizer for the depthwise convolution kernel.
* **pointwise\_regularizer**: Optional regularizer for the pointwise convolution kernel.
* **bias\_regularizer**: Optional regularizer for the bias vector.
* **activity\_regularizer**: Optional regularizer function for the output.
* **depthwise\_constraint**: Optional projection function to be applied to the depthwise kernel after being updated by an Optimizer (e.g. used for norm constraints or value constraints for layer weights). The function must take as input the unprojected variable and must return the projected variable (which must have the same shape). Constraints are not safe to use when doing asynchronous distributed training.
* **pointwise\_constraint**: Optional projection function to be applied to the pointwise kernel after being updated by an Optimizer.
* **bias\_constraint**: Optional projection function to be applied to the bias after being updated by an Optimizer.
* **trainable**: Boolean, if True the weights of this layer will be marked as trainable (and listed in layer.trainable\_weights).
* **name**: A string, the name of the layer.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=1,  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=1,  
    depth\_multiplier=1,  
    activation=None,  
    use\_bias=True,  
    depthwise\_initializer='glorot\_uniform',  
    pointwise\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    depthwise\_regularizer=None,  
    pointwise\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    depthwise\_constraint=None,  
    pointwise\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

# tf.keras.layers.SeparableConv2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv2D#top_of_page)
* [Class SeparableConv2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv2D#class_separableconv2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SeparableConv2D#__init__)

## Class SeparableConv2D

Depthwise separable 2D convolution.

### Aliases:

* Class tf.compat.v1.keras.layers.SeparableConv2D
* Class tf.compat.v1.keras.layers.SeparableConvolution2D
* Class tf.compat.v2.keras.layers.SeparableConv2D
* Class tf.compat.v2.keras.layers.SeparableConvolution2D
* Class tf.keras.layers.SeparableConv2D
* Class tf.keras.layers.SeparableConvolution2D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

Separable convolutions consist in first performing a depthwise spatial convolution (which acts on each input channel separately) followed by a pointwise convolution which mixes together the resulting output channels. The depth\_multiplier argument controls how many output channels are generated per input channel in the depthwise step.

Intuitively, separable convolutions can be understood as a way to factorize a convolution kernel into two smaller kernels, or as an extreme version of an Inception block.

#### Arguments:

* **filters**: Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
* **kernel\_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
* **strides**: An integer or tuple/list of 2 integers, specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions. Specifying any stride value != 1 is incompatible with specifying any dilation\_rate value != 1.
* **padding**: one of "valid" or "same" (case-insensitive).
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **dilation\_rate**: An integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Currently, specifying any dilation\_rate value != 1 is incompatible with specifying any strides value != 1.
* **depth\_multiplier**: The number of depthwise convolution output channels for each input channel. The total number of depthwise convolution output channels will be equal to filters\_in \* depth\_multiplier.
* **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **depthwise\_initializer**: Initializer for the depthwise kernel matrix.
* **pointwise\_initializer**: Initializer for the pointwise kernel matrix.
* **bias\_initializer**: Initializer for the bias vector.
* **depthwise\_regularizer**: Regularizer function applied to the depthwise kernel matrix.
* **pointwise\_regularizer**: Regularizer function applied to the pointwise kernel matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **depthwise\_constraint**: Constraint function applied to the depthwise kernel matrix.
* **pointwise\_constraint**: Constraint function applied to the pointwise kernel matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.

#### Input shape:

4D tensor with shape: (batch, channels, rows, cols) if data\_format='channels\_first' or 4D tensor with shape: (batch, rows, cols, channels) if data\_format='channels\_last'.

#### Output shape:

4D tensor with shape: (batch, filters, new\_rows, new\_cols) if data\_format='channels\_first' or 4D tensor with shape: (batch, new\_rows, new\_cols, filters) if data\_format='channels\_last'.rows and cols values might have changed due to padding.

## \_\_init\_\_

\_\_init\_\_(  
    filters,  
    kernel\_size,  
    strides=(1, 1),  
    padding='valid',  
    data\_format=None,  
    dilation\_rate=(1, 1),  
    depth\_multiplier=1,  
    activation=None,  
    use\_bias=True,  
    depthwise\_initializer='glorot\_uniform',  
    pointwise\_initializer='glorot\_uniform',  
    bias\_initializer='zeros',  
    depthwise\_regularizer=None,  
    pointwise\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    depthwise\_constraint=None,  
    pointwise\_constraint=None,  
    bias\_constraint=None,  
    \*\*kwargs  
)

tf.keras.layers.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/serialize#aliases)

Aliases:

* tf.compat.v1.keras.layers.serialize
* tf.compat.v2.keras.layers.serialize
* tf.keras.layers.serialize

tf.keras.layers.serialize(layer)

Defined in [python/keras/layers/serialization.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/serialization.py).

# tf.keras.layers.SimpleRNN

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNN#top_of_page)
* [Class SimpleRNN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNN#class_simplernn)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNN#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNN#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNN#properties)

## Class SimpleRNN

Fully-connected RNN where the output is to be fed back to input.

Inherits From: [RNN](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/RNN)

### Aliases:

* Class tf.compat.v1.keras.layers.SimpleRNN
* Class tf.compat.v2.keras.layers.SimpleRNN
* Class tf.keras.layers.SimpleRNN

Defined in [python/keras/layers/recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent.py).

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.
* **return\_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence.
* **return\_state**: Boolean. Whether to return the last state in addition to the output.
* **go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.
* **stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
* **unroll**: Boolean (default False). If True, the network will be unrolled, else a symbolic loop will be used. Unrolling can speed-up a RNN, although it tends to be more memory-intensive. Unrolling is only suitable for short sequences.

#### Call Arguments:

* **inputs**: A 3D tensor.
* **mask**: Binary tensor of shape (samples, timesteps) indicating whether a given timestep should be masked.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. This argument is passed to the cell when calling it. This is only relevant if dropout or recurrent\_dropout is used.
* **initial\_state**: List of initial state tensors to be passed to the first call of the cell.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    unroll=False,  
    \*\*kwargs  
)

## Properties

### activation

### bias\_constraint

### bias\_initializer

### bias\_regularizer

### dropout

### kernel\_constraint

### kernel\_initializer

### kernel\_regularizer

### recurrent\_constraint

### recurrent\_dropout

### recurrent\_initializer

### recurrent\_regularizer

### states

### units

### use\_bias

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states

reset\_states(states=None)

# tf.keras.layers.SimpleRNNCell

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNNCell#top_of_page)
* [Class SimpleRNNCell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNNCell#class_simplernncell)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNNCell#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNNCell#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SimpleRNNCell#methods)

## Class SimpleRNNCell

Cell class for SimpleRNN.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.SimpleRNNCell
* Class tf.compat.v2.keras.layers.SimpleRNNCell
* Class tf.keras.layers.SimpleRNNCell

Defined in [python/keras/layers/recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent.py).

#### Arguments:

* **units**: Positive integer, dimensionality of the output space.
* **activation**: Activation function to use. Default: hyperbolic tangent (tanh). If you pass None, no activation is applied (ie. "linear" activation: a(x) = x).
* **use\_bias**: Boolean, whether the layer uses a bias vector.
* **kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.
* **recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.
* **bias\_initializer**: Initializer for the bias vector.
* **kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.
* **recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.
* **bias\_regularizer**: Regularizer function applied to the bias vector.
* **kernel\_constraint**: Constraint function applied to the kernel weights matrix.
* **recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.
* **bias\_constraint**: Constraint function applied to the bias vector.
* **dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.
* **recurrent\_dropout**: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.

#### Call Arguments:

* **inputs**: A 2D tensor.
* **states**: List of state tensors corresponding to the previous timestep.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. Only relevant when dropout or recurrent\_dropout is used.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    activation='tanh',  
    use\_bias=True,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    dropout=0.0,  
    recurrent\_dropout=0.0,  
    \*\*kwargs  
)

## Methods

### get\_dropout\_mask\_for\_cell

get\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the dropout mask for RNN cell's input.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

### get\_recurrent\_dropout\_mask\_for\_cell

get\_recurrent\_dropout\_mask\_for\_cell(  
    inputs,  
    training,  
    count=1  
)

Get the recurrent dropout mask for RNN cell.

It will create mask based on context if there isn't any existing cached mask. If a new mask is generated, it will update the cache in the cell.

#### Args:

* **inputs**: the input tensor whose shape will be used to generate dropout mask.
* **training**: boolean tensor, whether its in training mode, dropout will be ignored in non-training mode.
* **count**: int, how many dropout mask will be generated. It is useful for cell that has internal weights fused together.

#### Returns:

List of mask tensor, generated or cached mask based on context.

### reset\_dropout\_mask

reset\_dropout\_mask()

Reset the cached dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

### reset\_recurrent\_dropout\_mask

reset\_recurrent\_dropout\_mask()

Reset the cached recurrent dropout masks if any.

This is important for the RNN layer to invoke this in it call() method so that the cached mask is cleared before calling the cell.call(). The mask should be cached across the timestep within the same batch, but shouldn't be cached between batches. Otherwise it will introduce unreasonable bias against certain index of data within the batch.

# tf.keras.layers.Softmax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Softmax#top_of_page)
* [Class Softmax](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Softmax#class_softmax)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Softmax#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Softmax#__init__)

## Class Softmax

Softmax activation function.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Softmax
* Class tf.compat.v2.keras.layers.Softmax
* Class tf.keras.layers.Softmax

Defined in [python/keras/layers/advanced\_activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/advanced_activations.py).

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as the input.

#### Arguments:

* **axis**: Integer, axis along which the softmax normalization is applied.

## \_\_init\_\_

\_\_init\_\_(  
    axis=-1,  
    \*\*kwargs  
)

# tf.keras.layers.SpatialDropout1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout1D#top_of_page)
* [Class SpatialDropout1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout1D#class_spatialdropout1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout1D#__init__)

## Class SpatialDropout1D

Spatial 1D version of Dropout.

Inherits From: [Dropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout)

### Aliases:

* Class tf.compat.v1.keras.layers.SpatialDropout1D
* Class tf.compat.v2.keras.layers.SpatialDropout1D
* Class tf.keras.layers.SpatialDropout1D

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

This version performs the same function as Dropout, however it drops entire 1D feature maps instead of individual elements. If adjacent frames within feature maps are strongly correlated (as is normally the case in early convolution layers) then regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. In this case, SpatialDropout1D will help promote independence between feature maps and should be used instead.

#### Arguments:

* **rate**: Float between 0 and 1. Fraction of the input units to drop.

#### Call Arguments:

* **inputs**: A 3D tensor.
* **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

#### Input shape:

3D tensor with shape: (samples, timesteps, channels)

#### Output shape:

Same as input.

#### References:

* [Efficient Object Localization Using Convolutional Networks](https://arxiv.org/abs/1411.4280)

## \_\_init\_\_

\_\_init\_\_(  
    rate,  
    \*\*kwargs  
)

# tf.keras.layers.SpatialDropout2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout2D#top_of_page)
* [Class SpatialDropout2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout2D#class_spatialdropout2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout2D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout2D#__init__)

## Class SpatialDropout2D

Spatial 2D version of Dropout.

Inherits From: [Dropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout)

### Aliases:

* Class tf.compat.v1.keras.layers.SpatialDropout2D
* Class tf.compat.v2.keras.layers.SpatialDropout2D
* Class tf.keras.layers.SpatialDropout2D

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

This version performs the same function as Dropout, however it drops entire 2D feature maps instead of individual elements. If adjacent pixels within feature maps are strongly correlated (as is normally the case in early convolution layers) then regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. In this case, SpatialDropout2D will help promote independence between feature maps and should be used instead.

#### Arguments:

* **rate**: Float between 0 and 1. Fraction of the input units to drop.
* **data\_format**: 'channels\_first' or 'channels\_last'. In 'channels\_first' mode, the channels dimension (the depth) is at index 1, in 'channels\_last' mode is it at index 3. It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Call Arguments:

* **inputs**: A 4D tensor.
* **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

#### Input shape:

4D tensor with shape: (samples, channels, rows, cols) if data\_format='channels\_first' or 4D tensor with shape: (samples, rows, cols, channels) if data\_format='channels\_last'.

#### Output shape:

Same as input.

#### References:

* [Efficient Object Localization Using Convolutional Networks](https://arxiv.org/abs/1411.4280)

## \_\_init\_\_

\_\_init\_\_(  
    rate,  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.SpatialDropout3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout3D#top_of_page)
* [Class SpatialDropout3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout3D#class_spatialdropout3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/SpatialDropout3D#__init__)

## Class SpatialDropout3D

Spatial 3D version of Dropout.

Inherits From: [Dropout](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dropout)

### Aliases:

* Class tf.compat.v1.keras.layers.SpatialDropout3D
* Class tf.compat.v2.keras.layers.SpatialDropout3D
* Class tf.keras.layers.SpatialDropout3D

Defined in [python/keras/layers/core.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/core.py).

This version performs the same function as Dropout, however it drops entire 3D feature maps instead of individual elements. If adjacent voxels within feature maps are strongly correlated (as is normally the case in early convolution layers) then regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. In this case, SpatialDropout3D will help promote independence between feature maps and should be used instead.

#### Arguments:

* **rate**: Float between 0 and 1. Fraction of the input units to drop.
* **data\_format**: 'channels\_first' or 'channels\_last'. In 'channels\_first' mode, the channels dimension (the depth) is at index 1, in 'channels\_last' mode is it at index 4. It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Call Arguments:

* **inputs**: A 5D tensor.
* **training**: Python boolean indicating whether the layer should behave in training mode (adding dropout) or in inference mode (doing nothing).

#### Input shape:

5D tensor with shape: (samples, channels, dim1, dim2, dim3) if data\_format='channels\_first' or 5D tensor with shape: (samples, dim1, dim2, dim3, channels) if data\_format='channels\_last'.

#### Output shape:

Same as input.

#### References:

* [Efficient Object Localization Using Convolutional Networks](https://arxiv.org/abs/1411.4280)

## \_\_init\_\_

\_\_init\_\_(  
    rate,  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.StackedRNNCells

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/StackedRNNCells#top_of_page)
* [Class StackedRNNCells](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/StackedRNNCells#class_stackedrnncells)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/StackedRNNCells#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/StackedRNNCells#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/StackedRNNCells#properties)

## Class StackedRNNCells

Wrapper allowing a stack of RNN cells to behave as a single cell.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.StackedRNNCells
* Class tf.compat.v2.keras.layers.StackedRNNCells
* Class tf.keras.layers.StackedRNNCells

Defined in [python/keras/layers/recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/recurrent.py).

Used to implement efficient stacked RNNs.

#### Arguments:

* **cells**: List of RNN cell instances.

#### Examples:

cells = [  
    keras.layers.LSTMCell(output\_dim),  
    keras.layers.LSTMCell(output\_dim),  
    keras.layers.LSTMCell(output\_dim),  
]  
  
inputs = keras.Input((timesteps, input\_dim))  
x = keras.layers.RNN(cells)(inputs)

## \_\_init\_\_

\_\_init\_\_(  
    cells,  
    \*\*kwargs  
)

## Properties

### output\_size

### state\_size

## Methods

### get\_initial\_state

get\_initial\_state(  
    inputs=None,  
    batch\_size=None,  
    dtype=None  
)

# tf.keras.layers.Subtract

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Subtract#top_of_page)
* [Class Subtract](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Subtract#class_subtract)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Subtract#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Subtract#__init__)

## Class Subtract

Layer that subtracts two inputs.

### Aliases:

* Class tf.compat.v1.keras.layers.Subtract
* Class tf.compat.v2.keras.layers.Subtract
* Class tf.keras.layers.Subtract

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

It takes as input a list of tensors of size 2, both of the same shape, and returns a single tensor, (inputs[0] - inputs[1]), also of the same shape.

#### Examples:

    import keras  
  
    input1 = keras.layers.Input(shape=(16,))  
    x1 = keras.layers.Dense(8, activation='relu')(input1)  
    input2 = keras.layers.Input(shape=(32,))  
    x2 = keras.layers.Dense(8, activation='relu')(input2)  
    # Equivalent to subtracted = keras.layers.subtract([x1, x2])  
    subtracted = keras.layers.Subtract()([x1, x2])  
  
    out = keras.layers.Dense(4)(subtracted)  
    model = keras.models.Model(inputs=[input1, input2], outputs=out)

## \_\_init\_\_

\_\_init\_\_(\*\*kwargs)

# tf.keras.layers.subtract

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/subtract#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/subtract#aliases)

Functional interface to the Subtract layer.

### Aliases:

* tf.compat.v1.keras.layers.subtract
* tf.compat.v2.keras.layers.subtract
* tf.keras.layers.subtract

tf.keras.layers.subtract(  
    inputs,  
    \*\*kwargs  
)

Defined in [python/keras/layers/merge.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/merge.py).

#### Arguments:

* **inputs**: A list of input tensors (exactly 2).
* **\*\*kwargs**: Standard layer keyword **Arguments**.

#### Returns:

A tensor, the difference of the inputs.

#### Examples:

    import keras  
  
    input1 = keras.layers.Input(shape=(16,))  
    x1 = keras.layers.Dense(8, activation='relu')(input1)  
    input2 = keras.layers.Input(shape=(32,))  
    x2 = keras.layers.Dense(8, activation='relu')(input2)  
    subtracted = keras.layers.subtract([x1, x2])  
  
    out = keras.layers.Dense(4)(subtracted)  
    model = keras.models.Model(inputs=[input1, input2], outputs=out)

# tf.keras.layers.ThresholdedReLU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ThresholdedReLU#top_of_page)
* [Class ThresholdedReLU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ThresholdedReLU#class_thresholdedrelu)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ThresholdedReLU#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ThresholdedReLU#__init__)

## Class ThresholdedReLU

Thresholded Rectified Linear Unit.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ThresholdedReLU
* Class tf.compat.v2.keras.layers.ThresholdedReLU
* Class tf.keras.layers.ThresholdedReLU

Defined in [python/keras/layers/advanced\_activations.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/advanced_activations.py).

#### It follows:

f(x) = x for x > theta, f(x) = 0 otherwise.

#### Input shape:

Arbitrary. Use the keyword argument input\_shape (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

#### Output shape:

Same shape as the input.

#### Arguments:

* **theta**: Float >= 0. Threshold location of activation.

## \_\_init\_\_

\_\_init\_\_(  
    theta=1.0,  
    \*\*kwargs  
)

# tf.keras.layers.TimeDistributed

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/TimeDistributed#top_of_page)
* [Class TimeDistributed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/TimeDistributed#class_timedistributed)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/TimeDistributed#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/TimeDistributed#__init__)

## Class TimeDistributed

This wrapper allows to apply a layer to every temporal slice of an input.

Inherits From: [Wrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper)

### Aliases:

* Class tf.compat.v1.keras.layers.TimeDistributed
* Class tf.compat.v2.keras.layers.TimeDistributed
* Class tf.keras.layers.TimeDistributed

Defined in [python/keras/layers/wrappers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/wrappers.py).

The input should be at least 3D, and the dimension of index one will be considered to be the temporal dimension.

Consider a batch of 32 samples, where each sample is a sequence of 10 vectors of 16 dimensions. The batch input shape of the layer is then (32, 10, 16), and the input\_shape, not including the samples dimension, is (10, 16).

You can then use TimeDistributed to apply a Dense layer to each of the 10 timesteps, independently:

# as the first layer in a model  
model = Sequential()  
model.add(TimeDistributed(Dense(8), input\_shape=(10, 16)))  
# now model.output\_shape == (None, 10, 8)

The output will then have shape (32, 10, 8).

In subsequent layers, there is no need for the input\_shape:

model.add(TimeDistributed(Dense(32)))  
# now model.output\_shape == (None, 10, 32)

The output will then have shape (32, 10, 32).

TimeDistributed can be used with arbitrary layers, not just Dense, for instance with a Conv2Dlayer:

model = Sequential()  
model.add(TimeDistributed(Conv2D(64, (3, 3)),  
                          input\_shape=(10, 299, 299, 3)))

#### Arguments:

* **layer**: a layer instance.

#### Call Arguments:

* **inputs**: Input tensor.
* **training**: Python boolean indicating whether the layer should behave in training mode or in inference mode. This argument is passed to the wrapped layer (only if the layer supports this argument).
* **mask**: Binary tensor of shape (samples, timesteps) indicating whether a given timestep should be masked. This argument is passed to the wrapped layer (only if the layer supports this argument).

#### Raises:

* **ValueError**: If not initialized with a Layer instance.

## \_\_init\_\_

\_\_init\_\_(  
    layer,  
    \*\*kwargs  
)

# tf.keras.layers.UpSampling1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling1D#top_of_page)
* [Class UpSampling1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling1D#class_upsampling1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling1D#__init__)

## Class UpSampling1D

Upsampling layer for 1D inputs.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.UpSampling1D
* Class tf.compat.v2.keras.layers.UpSampling1D
* Class tf.keras.layers.UpSampling1D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

Repeats each temporal step size times along the time axis.

#### Arguments:

* **size**: Integer. Upsampling factor.

#### Input shape:

3D tensor with shape: (batch, steps, features).

#### Output shape:

3D tensor with shape: (batch, upsampled\_steps, features).

## \_\_init\_\_

\_\_init\_\_(  
    size=2,  
    \*\*kwargs  
)

# tf.keras.layers.UpSampling2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling2D#top_of_page)
* [Class UpSampling2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling2D#class_upsampling2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling2D#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling2D#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling2D#__init__)

## Class UpSampling2D

Upsampling layer for 2D inputs.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.UpSampling2D
* Class tf.compat.v2.keras.layers.UpSampling2D
* Class tf.keras.layers.UpSampling2D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

### Used in the guide:

* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

Repeats the rows and columns of the data by size[0] and size[1] respectively.

#### Arguments:

* **size**: Int, or tuple of 2 integers. The upsampling factors for rows and columns.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **interpolation**: A string, one of nearest or bilinear.

#### Input shape:

4D tensor with shape: - If data\_format is "channels\_last": (batch, rows, cols, channels) - If data\_format is "channels\_first": (batch, channels, rows, cols)

#### Output shape:

4D tensor with shape: - If data\_format is "channels\_last": (batch, upsampled\_rows, upsampled\_cols, channels) - If data\_format is "channels\_first": (batch, channels, upsampled\_rows, upsampled\_cols)

## \_\_init\_\_

\_\_init\_\_(  
    size=(2, 2),  
    data\_format=None,  
    interpolation='nearest',  
    \*\*kwargs  
)

# tf.keras.layers.UpSampling3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling3D#top_of_page)
* [Class UpSampling3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling3D#class_upsampling3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/UpSampling3D#__init__)

## Class UpSampling3D

Upsampling layer for 3D inputs.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.UpSampling3D
* Class tf.compat.v2.keras.layers.UpSampling3D
* Class tf.keras.layers.UpSampling3D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

Repeats the 1st, 2nd and 3rd dimensions of the data by size[0], size[1] and size[2]respectively.

#### Arguments:

* **size**: Int, or tuple of 3 integers. The upsampling factors for dim1, dim2 and dim3.
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

5D tensor with shape: - If data\_format is "channels\_last": (batch, dim1, dim2, dim3, channels) - If data\_format is "channels\_first": (batch, channels, dim1, dim2, dim3)

#### Output shape:

5D tensor with shape: - If data\_format is "channels\_last": (batch, upsampled\_dim1, upsampled\_dim2, upsampled\_dim3, channels) - If data\_format is "channels\_first": (batch, channels, upsampled\_dim1, upsampled\_dim2, upsampled\_dim3)

## \_\_init\_\_

# \_\_init\_\_(     size=(2, 2, 2),     data\_format=None,     \*\*kwargs ) tf.keras.layers.Wrapper

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper#top_of_page)
* [Class Wrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper#class_wrapper)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Wrapper#__init__)

## Class Wrapper

Abstract wrapper base class.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.Wrapper
* Class tf.compat.v2.keras.layers.Wrapper
* Class tf.keras.layers.Wrapper

Defined in [python/keras/layers/wrappers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/wrappers.py).

Wrappers take another layer and augment it in various ways. Do not use this class as a layer, it is only an abstract base class. Two usable wrappers are the TimeDistributed and Bidirectionalwrappers.

#### Arguments:

* **layer**: The layer to be wrapped.

## \_\_init\_\_

\_\_init\_\_(  
    layer,  
    \*\*kwargs  
)

# tf.keras.layers.ZeroPadding1D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding1D#top_of_page)
* [Class ZeroPadding1D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding1D#class_zeropadding1d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding1D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding1D#__init__)

## Class ZeroPadding1D

Zero-padding layer for 1D input (e.g. temporal sequence).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ZeroPadding1D
* Class tf.compat.v2.keras.layers.ZeroPadding1D
* Class tf.keras.layers.ZeroPadding1D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

#### Arguments:

* **padding**: Int, or tuple of int (length 2), or dictionary.
  + If int: How many zeros to add at the beginning and end of the padding dimension (axis 1).
  + If tuple of int (length 2): How many zeros to add at the beginning and at the end of the padding dimension ((left\_pad, right\_pad)).

#### Input shape:

3D tensor with shape (batch, axis\_to\_pad, features)

#### Output shape:

3D tensor with shape (batch, padded\_axis, features)

## \_\_init\_\_

\_\_init\_\_(  
    padding=1,  
    \*\*kwargs  
)

# tf.keras.layers.ZeroPadding2D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding2D#top_of_page)
* [Class ZeroPadding2D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding2D#class_zeropadding2d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding2D#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding2D#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding2D#__init__)

## Class ZeroPadding2D

Zero-padding layer for 2D input (e.g. picture).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ZeroPadding2D
* Class tf.compat.v2.keras.layers.ZeroPadding2D
* Class tf.keras.layers.ZeroPadding2D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

This layer can add rows and columns of zeros at the top, bottom, left and right side of an image tensor.

#### Arguments:

* **padding**: Int, or tuple of 2 ints, or tuple of 2 tuples of 2 ints.
  + If int: the same symmetric padding is applied to height and width.
  + If tuple of 2 ints: interpreted as two different symmetric padding values for height and width: (symmetric\_height\_pad, symmetric\_width\_pad).
  + If tuple of 2 tuples of 2 ints: interpreted as ((top\_pad, bottom\_pad), (left\_pad, right\_pad))
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, height, width, channels) while channels\_first corresponds to inputs with shape(batch, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

4D tensor with shape: - If data\_format is "channels\_last": (batch, rows, cols, channels) - If data\_format is "channels\_first": (batch, channels, rows, cols)

#### Output shape:

4D tensor with shape: - If data\_format is "channels\_last": (batch, padded\_rows, padded\_cols, channels) - If data\_format is "channels\_first": (batch, channels, padded\_rows, padded\_cols)

## \_\_init\_\_

\_\_init\_\_(  
    padding=(1, 1),  
    data\_format=None,  
    \*\*kwargs  
)

# tf.keras.layers.ZeroPadding3D

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding3D#top_of_page)
* [Class ZeroPadding3D](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding3D#class_zeropadding3d)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding3D#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/ZeroPadding3D#__init__)

## Class ZeroPadding3D

Zero-padding layer for 3D data (spatial or spatio-temporal).

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.layers.ZeroPadding3D
* Class tf.compat.v2.keras.layers.ZeroPadding3D
* Class tf.keras.layers.ZeroPadding3D

Defined in [python/keras/layers/convolutional.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/convolutional.py).

#### Arguments:

* **padding**: Int, or tuple of 3 ints, or tuple of 3 tuples of 2 ints.
  + If int: the same symmetric padding is applied to height and width.
  + If tuple of 3 ints: interpreted as two different symmetric padding values for height and width: (symmetric\_dim1\_pad, symmetric\_dim2\_pad, symmetric\_dim3\_pad).
  + If tuple of 3 tuples of 2 ints: interpreted as ((left\_dim1\_pad, right\_dim1\_pad), (left\_dim2\_pad, right\_dim2\_pad), (left\_dim3\_pad, right\_dim3\_pad))
* **data\_format**: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_firstcorresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

#### Input shape:

5D tensor with shape: - If data\_format is "channels\_last": (batch, first\_axis\_to\_pad, second\_axis\_to\_pad, third\_axis\_to\_pad, depth) - If data\_format is "channels\_first":(batch, depth, first\_axis\_to\_pad, second\_axis\_to\_pad, third\_axis\_to\_pad)

#### Output shape:

5D tensor with shape: - If data\_format is "channels\_last": (batch, first\_padded\_axis, second\_padded\_axis, third\_axis\_to\_pad, depth) - If data\_format is "channels\_first":(batch, depth, first\_padded\_axis, second\_padded\_axis, third\_axis\_to\_pad)

## \_\_init\_\_

\_\_init\_\_(  
    padding=(1, 1, 1),  
    data\_format=None,  
    \*\*kwargs  
)

Module: tf.keras.losses

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses#functions)

Built-in loss functions.

Aliases:

* Module tf.compat.v2.keras.losses
* Module tf.compat.v2.losses
* Module tf.keras.losses
* Module tf.losses

Defined in [python/keras/api/\_v2/keras/losses/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/losses/__init__.py).

Classes

[class BinaryCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/BinaryCrossentropy): Computes the cross-entropy loss between true labels and predicted labels.

[class CategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalCrossentropy): Computes the crossentropy loss between the labels and predictions.

[class CategoricalHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge): Computes the categorical hinge loss between y\_true and y\_pred.

[class CosineSimilarity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CosineSimilarity): Computes the cosine similarity between y\_true and y\_pred.

[class Hinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Hinge): Computes the hinge loss between y\_true and y\_pred.

[class Huber](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Huber): Computes the Huber loss between y\_true and y\_pred.

[class KLDivergence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLDivergence): Computes Kullback Leibler divergence loss between y\_true and y\_pred.

[class LogCosh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/LogCosh): Computes the logarithm of the hyperbolic cosine of the prediction error.

[class Loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss): Loss base class.

[class MeanAbsoluteError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsoluteError): Computes the mean of absolute difference between labels and predictions.

[class MeanAbsolutePercentageError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsolutePercentageError): Computes the mean absolute percentage error between y\_true and y\_pred.

[class MeanSquaredError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredError): Computes the mean of squares of errors between labels and predictions.

[class MeanSquaredLogarithmicError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError): Computes the mean squared logarithmic error between y\_true and y\_pred.

[class Poisson](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Poisson): Computes the Poisson loss between y\_true and y\_pred.

[class Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction): Types of loss reduction.

[class SparseCategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy): Computes the crossentropy loss between the labels and predictions.

[class SquaredHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SquaredHinge): Computes the squared hinge loss between y\_true and y\_pred.

Functions

[KLD(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD)

[MAE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE)

[MAPE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE)

[MSE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE)

[MSLE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE)

[binary\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/binary_crossentropy)

[categorical\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/categorical_crossentropy): Computes the categorical crossentropy loss.

[categorical\_hinge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/categorical_hinge)

[cosine\_similarity(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/cosine_similarity): Computes the cosine similarity between labels and predictions.

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/deserialize)

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/get)

[hinge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/hinge): Computes the hinge loss between y\_true and y\_pred.

[kld(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD)

[kullback\_leibler\_divergence(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD)

[logcosh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/logcosh): Logarithm of the hyperbolic cosine of the prediction error.

[mae(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE)

[mape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE)

[mean\_absolute\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE)

[mean\_absolute\_percentage\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE)

[mean\_squared\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE)

[mean\_squared\_logarithmic\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE)

[mse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE)

[msle(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE)

[poisson(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/poisson): Computes the Poisson loss between y\_true and y\_pred.

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/serialize)

[sparse\_categorical\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/sparse_categorical_crossentropy)

[squared\_hinge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/squared_hinge): Computes the squared hinge loss between y\_true and y\_pred.

Module: tf.compat.v1.losses

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses#top_of_page)

[Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses#classes)

[Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses#functions)

Loss operations for use in neural networks.

**Note:** All the losses are added to the **GraphKeys.LOSSES** collection by default.

Classes

[class Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction): Types of loss reduction.

Functions

[absolute\_difference(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/absolute_difference): Adds an Absolute Difference loss to the training procedure.

[add\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/add_loss): Adds a externally defined loss to the collection of losses.

[compute\_weighted\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/compute_weighted_loss): Computes the weighted loss.

[cosine\_distance(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/cosine_distance): Adds a cosine-distance loss to the training procedure. (deprecated arguments)

[get\_losses(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/get_losses): Gets the list of losses from the loss\_collection.

[get\_regularization\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/get_regularization_loss): Gets the total regularization loss.

[get\_regularization\_losses(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/get_regularization_losses): Gets the list of regularization losses.

[get\_total\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/get_total_loss): Returns a tensor whose value represents the total loss.

[hinge\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/hinge_loss): Adds a hinge loss to the training procedure.

[huber\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/huber_loss): Adds a Huber Loss term to the training procedure.

[log\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/log_loss): Adds a Log Loss term to the training procedure.

[mean\_pairwise\_squared\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/mean_pairwise_squared_error): Adds a pairwise-errors-squared loss to the training procedure.

[mean\_squared\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/mean_squared_error): Adds a Sum-of-Squares loss to the training procedure.

[sigmoid\_cross\_entropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/sigmoid_cross_entropy): Creates a cross-entropy loss using tf.nn.sigmoid\_cross\_entropy\_with\_logits.

[softmax\_cross\_entropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/softmax_cross_entropy): Creates a cross-entropy loss using tf.nn.softmax\_cross\_entropy\_with\_logits\_v2.

[sparse\_softmax\_cross\_entropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/sparse_softmax_cross_entropy): Cross-entropy loss using [tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/sparse_softmax_cross_entropy_with_logits).

# tf.compat.v1.losses.absolute\_difference

Adds an Absolute Difference loss to the training procedure.

tf.compat.v1.losses.absolute\_difference(  
    labels,  
    predictions,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a Tensor of shape [batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the weights vector. If the shape of weightsmatches the shape of predictions, then the loss of each measurable element of predictions is scaled by the corresponding value of weights.

#### Args:

**labels**: The ground truth output tensor, same dimensions as 'predictions'.

**predictions**: The predicted outputs.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which this loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shape of predictions doesn't match that of labels or if the shape of weights is invalid or if labels or predictions is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.add\_loss

Adds a externally defined loss to the collection of losses.

tf.compat.v1.losses.add\_loss(  
    loss,  
    loss\_collection=tf.GraphKeys.LOSSES  
)

Defined in [python/ops/losses/util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/util.py).

#### Args:

**loss**: A loss Tensor.

**loss\_collection**: Optional collection to add the loss to.

# tf.compat.v1.losses.compute\_weighted\_loss

Computes the weighted loss.

tf.compat.v1.losses.compute\_weighted\_loss(  
    losses,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

#### Args:

**losses**: Tensor of shape [batch\_size, d1, ... dN].

**weights**: Optional Tensor whose rank is either 0, or the same rank as losses, and must be broadcastable to losses (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**scope**: the scope for the operations performed in computing the loss.

**loss\_collection**: the loss will be added to these collections.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss Tensor of the same type as losses. If reduction is NONE, this has the same shape as losses; otherwise, it is scalar.

#### Raises:

**ValueError**: If weights is None or the shape is not compatible with losses, or if the number of dimensions (rank) of either losses or weights is missing.

#### Note:

When calculating the gradient of a weighted loss contributions from both losses and weights are considered. If your weights depend on some model parameters but you do not want this to affect the loss gradient, you need to apply [tf.stop\_gradient](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/stop_gradient) to weights before passing them to compute\_weighted\_loss.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.cosine\_distance

Adds a cosine-distance loss to the training procedure. (deprecated arguments)

tf.compat.v1.losses.cosine\_distance(  
    labels,  
    predictions,  
    axis=None,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS,  
    dim=None  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dim)**. They will be removed in a future version. Instructions for updating: dim is deprecated, use axis instead

Note that the function assumes that predictions and labels are already unit-normalized.

#### Args:

**labels**: Tensor whose shape matches 'predictions'

**predictions**: An arbitrary matrix.

**axis**: The dimension along which the cosine distance is computed.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which this loss will be added.

**reduction**: Type of reduction to apply to loss.

**dim**: The old (deprecated) name for axis.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If predictions shape doesn't match labels shape, or axis, labels, predictions or weights is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.get\_losses

Gets the list of losses from the loss\_collection.

tf.compat.v1.losses.get\_losses(  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES  
)

Defined in [python/ops/losses/util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/util.py).

#### Args:

**scope**: An optional scope name for filtering the losses to return.

**loss\_collection**: Optional losses collection.

#### Returns:

a list of loss tensors.

# tf.compat.v1.losses.get\_regularization\_loss

Gets the total regularization loss.

tf.compat.v1.losses.get\_regularization\_loss(  
    scope=None,  
    name='total\_regularization\_loss'  
)

Defined in [python/ops/losses/util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/util.py).

#### Args:

**scope**: An optional scope name for filtering the losses to return.

**name**: The name of the returned tensor.

#### Returns:

A scalar regularization loss.

# tf.compat.v1.losses.get\_regularization\_losses

Gets the list of regularization losses.

tf.compat.v1.losses.get\_regularization\_losses(scope=None)

Defined in [python/ops/losses/util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/util.py).

#### Args:

**scope**: An optional scope name for filtering the losses to return.

#### Returns:

A list of regularization losses as Tensors.

# tf.compat.v1.losses.get\_total\_loss

Returns a tensor whose value represents the total loss.

tf.compat.v1.losses.get\_total\_loss(  
    add\_regularization\_losses=True,  
    name='total\_loss'  
)

Defined in [python/ops/losses/util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/util.py).

In particular, this adds any losses you have added with tf.add\_loss() to any regularization losses that have been added by regularization parameters on layers constructors e.g. tf.layers. Be very sure to use this if you are constructing a loss\_op manually. Otherwise regularization arguments on tf.layers methods will not function.

#### Args:

**add\_regularization\_losses**: A boolean indicating whether or not to use the regularization losses in the sum.

**name**: The name of the returned tensor.

#### Returns:

A Tensor whose value represents the total loss.

#### Raises:

**ValueError**: if losses is not iterable.

# tf.compat.v1.losses.hinge\_loss

Adds a hinge loss to the training procedure.

tf.compat.v1.losses.hinge\_loss(  
    labels,  
    logits,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

#### Args:

**labels**: The ground truth output tensor. Its shape should match the shape of logits. The values of the tensor are expected to be 0.0 or 1.0. Internally the {0,1} labels are converted to {-1,1} when calculating the hinge loss.

**logits**: The logits, a float tensor. Note that logits are assumed to be unbounded and 0-centered. A value > 0 (resp. < 0) is considered a positive (resp. negative) binary prediction.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shapes of logits and labels don't match or if labels or logits is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.huber\_loss

Adds a Huber Loss term to the training procedure.

tf.compat.v1.losses.huber\_loss(  
    labels,  
    predictions,  
    weights=1.0,  
    delta=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

For each value x in error=labels-predictions, the following is calculated:

  0.5 \* x^2                  if |x| <= d  
  0.5 \* d^2 + d \* (|x| - d)  if |x| > d

where d is delta.

See: https://en.wikipedia.org/wiki/Huber\_loss

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of size [batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the weights vector. If the shape of weightsmatches the shape of predictions, then the loss of each measurable element of predictions is scaled by the corresponding value of weights.

#### Args:

**labels**: The ground truth output tensor, same dimensions as 'predictions'.

**predictions**: The predicted outputs.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**delta**: float, the point where the huber loss function changes from a quadratic to linear.

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shape of predictions doesn't match that of labels or if the shape of weights is invalid. Also if labels or predictions is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.log\_loss

Adds a Log Loss term to the training procedure.

tf.compat.v1.losses.log\_loss(  
    labels,  
    predictions,  
    weights=1.0,  
    epsilon=1e-07,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of size [batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the weights vector. If the shape of weightsmatches the shape of predictions, then the loss of each measurable element of predictions is scaled by the corresponding value of weights.

#### Args:

**labels**: The ground truth output tensor, same dimensions as 'predictions'.

**predictions**: The predicted outputs.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**epsilon**: A small increment to add to avoid taking a log of zero.

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shape of predictions doesn't match that of labels or if the shape of weights is invalid. Also if labels or predictions is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.mean\_pairwise\_squared\_error

Adds a pairwise-errors-squared loss to the training procedure.

tf.compat.v1.losses.mean\_pairwise\_squared\_error(  
    labels,  
    predictions,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

Unlike mean\_squared\_error, which is a measure of the differences between corresponding elements of predictions and labels, mean\_pairwise\_squared\_error is a measure of the differences between pairs of corresponding elements of predictions and labels.

For example, if labels=[a, b, c] and predictions=[x, y, z], there are three pairs of differences are summed to compute the loss: loss = [ ((a-b) - (x-y)).^2 + ((a-c) - (x-z)).^2 + ((b-c) - (y-z)).^2 ] / 3

Note that since the inputs are of shape [batch\_size, d0, ... dN], the corresponding pairs are computed within each batch sample but not across samples within a batch. For example, if predictions represents a batch of 16 grayscale images of dimension [batch\_size, 100, 200], then the set of pairs is drawn from each image, but not across images.

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of size [batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the weights vector.

#### Args:

**labels**: The ground truth output tensor, whose shape must match the shape of predictions.

**predictions**: The predicted outputs, a tensor of size [batch\_size, d0, .. dN] where N+1 is the total number of dimensions in predictions.

**weights**: Coefficients for the loss a scalar, a tensor of shape [batch\_size] or a tensor whose shape matches predictions.

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

#### Returns:

A scalar Tensor that returns the weighted loss.

#### Raises:

**ValueError**: If the shape of predictions doesn't match that of labels or if the shape of weights is invalid. Also if labels or predictions is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.mean\_squared\_error

Adds a Sum-of-Squares loss to the training procedure.

tf.compat.v1.losses.mean\_squared\_error(  
    labels,  
    predictions,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of size [batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the weights vector. If the shape of weightsmatches the shape of predictions, then the loss of each measurable element of predictions is scaled by the corresponding value of weights.

#### Args:

**labels**: The ground truth output tensor, same dimensions as 'predictions'.

**predictions**: The predicted outputs.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shape of predictions doesn't match that of labels or if the shape of weights is invalid. Also if labels or predictions is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.Reduction

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction#top_of_page)

[Class Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction#class_reduction)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction#methods)

[all](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction#all)

[validate](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction#validate)

[Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/losses/Reduction#class_members)

## Class Reduction

Types of loss reduction.

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

Contains the following values:

NONE: Un-reduced weighted losses with the same shape as input.

SUM: Scalar sum of weighted losses.

MEAN: Scalar SUM divided by sum of weights. DEPRECATED.

SUM\_OVER\_BATCH\_SIZE: Scalar SUM divided by number of elements in losses.

SUM\_OVER\_NONZERO\_WEIGHTS: Scalar SUM divided by number of non-zero weights. DEPRECATED.

SUM\_BY\_NONZERO\_WEIGHTS: Same as SUM\_OVER\_NONZERO\_WEIGHTS. DEPRECATED.

## Methods

### all

@classmethod  
all(cls)

### validate

@classmethod  
validate(  
    cls,  
    key  
)

## Class Members

MEAN = 'weighted\_mean'

NONE = 'none'

SUM = 'weighted\_sum'

SUM\_BY\_NONZERO\_WEIGHTS = 'weighted\_sum\_by\_nonzero\_weights'

SUM\_OVER\_BATCH\_SIZE = 'weighted\_sum\_over\_batch\_size'

SUM\_OVER\_NONZERO\_WEIGHTS = 'weighted\_sum\_by\_nonzero\_weights'

# tf.compat.v1.losses.sigmoid\_cross\_entropy

Creates a cross-entropy loss using tf.nn.sigmoid\_cross\_entropy\_with\_logits.

tf.compat.v1.losses.sigmoid\_cross\_entropy(  
    multi\_class\_labels,  
    logits,  
    weights=1.0,  
    label\_smoothing=0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of shape [batch\_size], then the loss weights apply to each corresponding sample.

If label\_smoothing is nonzero, smooth the labels towards 1/2:

new\_multiclass\_labels = multiclass\_labels \* (1 - label\_smoothing)  
                        + 0.5 \* label\_smoothing

#### Args:

**multi\_class\_labels**: [batch\_size, num\_classes] target integer labels in {0, 1}.

**logits**: Float [batch\_size, num\_classes] logits outputs of the network.

**weights**: Optional Tensor whose rank is either 0, or the same rank as labels, and must be broadcastable to labels (i.e., all dimensions must be either 1, or the same as the corresponding losses dimension).

**label\_smoothing**: If greater than 0 then smooth the labels.

**scope**: The scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss Tensor of the same type as logits. If reduction is NONE, this has the same shape as logits; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shape of logits doesn't match that of multi\_class\_labels or if the shape of weights is invalid, or if weights is None. Also if multi\_class\_labels or logits is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.softmax\_cross\_entropy

Creates a cross-entropy loss using tf.nn.softmax\_cross\_entropy\_with\_logits\_v2.

tf.compat.v1.losses.softmax\_cross\_entropy(  
    onehot\_labels,  
    logits,  
    weights=1.0,  
    label\_smoothing=0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of shape [batch\_size], then the loss weights apply to each corresponding sample.

If label\_smoothing is nonzero, smooth the labels towards 1/num\_classes: new\_onehot\_labels = onehot\_labels \* (1 - label\_smoothing) + label\_smoothing / num\_classes

Note that onehot\_labels and logits must have the same shape, e.g. [batch\_size, num\_classes]. The shape of weights must be broadcastable to loss, whose shape is decided by the shape of logits. In case the shape of logits is [batch\_size, num\_classes], loss is a Tensorof shape [batch\_size].

#### Args:

**onehot\_labels**: One-hot-encoded labels.

**logits**: Logits outputs of the network.

**weights**: Optional Tensor that is broadcastable to loss.

**label\_smoothing**: If greater than 0 then smooth the labels.

**scope**: the scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss Tensor of the same type as logits. If reduction is NONE, this has shape [batch\_size]; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shape of logits doesn't match that of onehot\_labels or if the shape of weights is invalid or if weights is None. Also if onehot\_labels or logits is None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.compat.v1.losses.sparse\_softmax\_cross\_entropy

Cross-entropy loss using [tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/sparse_softmax_cross_entropy_with_logits).

tf.compat.v1.losses.sparse\_softmax\_cross\_entropy(  
    labels,  
    logits,  
    weights=1.0,  
    scope=None,  
    loss\_collection=tf.GraphKeys.LOSSES,  
    reduction=Reduction.SUM\_BY\_NONZERO\_WEIGHTS  
)

Defined in [python/ops/losses/losses\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/losses_impl.py).

weights acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If weights is a tensor of shape [batch\_size], then the loss weights apply to each corresponding sample.

#### Args:

**labels**: Tensor of shape [d\_0, d\_1, ..., d\_{r-1}] (where r is rank of labels and result) and dtype int32 or int64. Each entry in labels must be an index in [0, num\_classes). Other values will raise an exception when this op is run on CPU, and return NaNfor corresponding loss and gradient rows on GPU.

**logits**: Unscaled log probabilities of shape [d\_0, d\_1, ..., d\_{r-1}, num\_classes] and dtype float16, float32 or float64.

**weights**: Coefficients for the loss. This must be scalar or broadcastable to labels (i.e. same rank and each dimension is either 1 or the same).

**scope**: the scope for the operations performed in computing the loss.

**loss\_collection**: collection to which the loss will be added.

**reduction**: Type of reduction to apply to loss.

#### Returns:

Weighted loss Tensor of the same type as logits. If reduction is NONE, this has the same shape as labels; otherwise, it is scalar.

#### Raises:

**ValueError**: If the shapes of logits, labels, and weights are incompatible, or if any of them are None.

#### Eager Compatibility

The loss\_collection argument is ignored when executing eagerly. Consider holding on to the return value or collecting losses via a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model).

# tf.keras.losses.BinaryCrossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/BinaryCrossentropy#top_of_page)
* [Class BinaryCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/BinaryCrossentropy#class_binarycrossentropy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/BinaryCrossentropy#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/BinaryCrossentropy#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/BinaryCrossentropy#__init__)

## Class BinaryCrossentropy

Computes the cross-entropy loss between true labels and predicted labels.

### Aliases:

* Class tf.compat.v1.keras.losses.BinaryCrossentropy
* Class tf.compat.v2.keras.losses.BinaryCrossentropy
* Class tf.compat.v2.losses.BinaryCrossentropy
* Class tf.keras.losses.BinaryCrossentropy
* Class tf.losses.BinaryCrossentropy

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

### Used in the tutorials:

* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

Use this cross-entropy loss when there are only two label classes (assumed to be 0 and 1). For each example, there should be a single floating-point value per prediction.

In the snippet below, each of the four examples has only a single floating-pointing value, and both y\_pred and y\_true have the shape [batch\_size].

#### Usage:

bce = tf.keras.losses.BinaryCrossentropy()  
loss = bce([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Loss: ', loss.numpy())  # Loss: 11.522857

Usage with the [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.BinaryCrossentropy())

#### Args:

* **from\_logits**: Whether to interpret y\_pred as a tensor of [logit](https://en.wikipedia.org/wiki/Logit) values. By default, we assume that y\_pred contains probabilities (i.e., values in [0, 1]).
* **label\_smoothing**: Float in [0, 1]. When 0, no smoothing occurs. When > 0, we compute the loss between the predicted labels and a smoothed version of the true labels, where the smoothing squeezes the labels towards 0.5. Larger values of label\_smoothing correspond to heavier smoothing.
* **reduction**: (Optional) Type of [tf.keras.losses.Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction) to apply to loss. Default value is AUTO. AUTO indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, using AUTO or SUM\_OVER\_BATCH\_SIZE will raise an error. Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.
* **name**: (Optional) Name for the op.

## \_\_init\_\_

\_\_init\_\_(  
    from\_logits=False,  
    label\_smoothing=0,  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='binary\_crossentropy'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.binary\_crossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/binary_crossentropy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/binary_crossentropy#aliases)

Aliases:

* tf.compat.v1.keras.losses.binary\_crossentropy
* tf.compat.v1.keras.metrics.binary\_crossentropy
* tf.compat.v2.keras.losses.binary\_crossentropy
* tf.compat.v2.keras.metrics.binary\_crossentropy
* tf.compat.v2.losses.binary\_crossentropy
* tf.compat.v2.metrics.binary\_crossentropy
* tf.keras.losses.binary\_crossentropy
* tf.keras.metrics.binary\_crossentropy
* tf.losses.binary\_crossentropy
* tf.metrics.binary\_crossentropy

tf.keras.losses.binary\_crossentropy(  
    y\_true,  
    y\_pred,  
    from\_logits=False,  
    label\_smoothing=0  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.CategoricalCrossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalCrossentropy#top_of_page)
* [Class CategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalCrossentropy#class_categoricalcrossentropy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalCrossentropy#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalCrossentropy#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalCrossentropy#__init__)

## Class CategoricalCrossentropy

Computes the crossentropy loss between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.losses.CategoricalCrossentropy
* Class tf.compat.v2.keras.losses.CategoricalCrossentropy
* Class tf.compat.v2.losses.CategoricalCrossentropy
* Class tf.keras.losses.CategoricalCrossentropy
* Class tf.losses.CategoricalCrossentropy

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

Use this crossentropy loss function when there are two or more label classes. We expect labels to be provided in a one\_hot representation. If you want to provide labels as integers, please use SparseCategoricalCrossentropy loss. There should be # classes floating point values per feature.

In the snippet below, there is # classes floating pointing values per example. The shape of both y\_pred and y\_true are [batch\_size, num\_classes].

#### Usage:

cce = tf.keras.losses.CategoricalCrossentropy()  
loss = cce(  
  [[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]],  
  [[.9, .05, .05], [.5, .89, .6], [.05, .01, .94]])  
print('Loss: ', loss.numpy())  # Loss: 0.3239

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.CategoricalCrossentropy())

#### Args:

* **from\_logits**: Whether y\_pred is expected to be a logits tensor. By default, we assume that y\_pred encodes a probability distribution.
* **label\_smoothing**: Float in [0, 1]. When > 0, label values are smoothed, meaning the confidence on label values are relaxed. e.g. label\_smoothing=0.2 means that we will use a value of 0.1for label 0 and 0.9 for label 1"
* **reduction**: (Optional) Type of [tf.keras.losses.Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction) to apply to loss. Default value is AUTO. AUTO indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, using AUTO or SUM\_OVER\_BATCH\_SIZE will raise an error. Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.
* **name**: Optional name for the op.

## \_\_init\_\_

\_\_init\_\_(  
    from\_logits=False,  
    label\_smoothing=0,  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='categorical\_crossentropy'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.CategoricalHinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#top_of_page)
* [Class CategoricalHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#class_categoricalhinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#methods)

## Class CategoricalHinge

Computes the categorical hinge loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.CategoricalHinge
* Class tf.compat.v2.keras.losses.CategoricalHinge
* Class tf.compat.v2.losses.CategoricalHinge
* Class tf.keras.losses.CategoricalHinge
* Class tf.losses.CategoricalHinge

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

#### Usage:

ch = tf.keras.losses.CategoricalHinge()  
loss = ch([0., 1., 1.], [1., 0., 1.])  
print('Loss: ', loss.numpy())  # Loss: 1.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.CategoricalHinge())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='categorical\_hinge'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.CategoricalHinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#top_of_page)
* [Class CategoricalHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#class_categoricalhinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CategoricalHinge#methods)

## Class CategoricalHinge

Computes the categorical hinge loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.CategoricalHinge
* Class tf.compat.v2.keras.losses.CategoricalHinge
* Class tf.compat.v2.losses.CategoricalHinge
* Class tf.keras.losses.CategoricalHinge
* Class tf.losses.CategoricalHinge

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

#### Usage:

ch = tf.keras.losses.CategoricalHinge()  
loss = ch([0., 1., 1.], [1., 0., 1.])  
print('Loss: ', loss.numpy())  # Loss: 1.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.CategoricalHinge())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='categorical\_hinge'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.categorical\_hinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/categorical_hinge#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/categorical_hinge#aliases)

Aliases:

* tf.compat.v1.keras.losses.categorical\_hinge
* tf.compat.v2.keras.losses.categorical\_hinge
* tf.compat.v2.losses.categorical\_hinge
* tf.keras.losses.categorical\_hinge
* tf.losses.categorical\_hinge

tf.keras.losses.categorical\_hinge(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.CosineSimilarity

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CosineSimilarity#top_of_page)
* [Class CosineSimilarity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CosineSimilarity#class_cosinesimilarity)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CosineSimilarity#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CosineSimilarity#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/CosineSimilarity#methods)

## Class CosineSimilarity

Computes the cosine similarity between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.CosineSimilarity
* Class tf.compat.v2.keras.losses.CosineSimilarity
* Class tf.compat.v2.losses.CosineSimilarity
* Class tf.keras.losses.CosineSimilarity
* Class tf.losses.CosineSimilarity

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

#### Usage:

cosine\_loss = tf.keras.losses.CosineSimilarity(axis=1)  
loss = cosine\_loss([[0., 1.], [1., 1.]], [[1., 0.], [1., 1.]])  
# l2\_norm(y\_true) = [[0., 1.], [1./1.414], 1./1.414]]]  
# l2\_norm(y\_pred) = [[1., 0.], [1./1.414], 1./1.414]]]  
# l2\_norm(y\_true) . l2\_norm(y\_pred) = [[0., 0.], [0.5, 0.5]]  
# loss = mean(sum(l2\_norm(y\_true) . l2\_norm(y\_pred), axis=1))  
       = ((0. + 0.) +  (0.5 + 0.5)) / 2  
  
print('Loss: ', loss.numpy())  # Loss: 0.5

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.CosineSimilarity(axis=1))

#### Args:

* **axis**: (Optional) Defaults to -1. The dimension along which the cosine similarity is computed.
* **reduction**: (Optional) Type of [tf.keras.losses.Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction) to apply to loss. Default value is AUTO. AUTO indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, using AUTO or SUM\_OVER\_BATCH\_SIZE will raise an error. Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.
* **name**: Optional name for the op.

## \_\_init\_\_

\_\_init\_\_(  
    axis=-1,  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='cosine\_similarity'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.cosine\_similarity

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/cosine_similarity#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/cosine_similarity#aliases)

Computes the cosine similarity between labels and predictions.

Aliases:

* tf.compat.v1.keras.losses.cosine
* tf.compat.v1.keras.losses.cosine\_proximity
* tf.compat.v1.keras.losses.cosine\_similarity
* tf.compat.v1.keras.metrics.cosine
* tf.compat.v1.keras.metrics.cosine\_proximity
* tf.compat.v2.keras.losses.cosine\_similarity
* tf.compat.v2.losses.cosine\_similarity
* tf.keras.losses.cosine\_similarity
* tf.losses.cosine\_similarity

tf.keras.losses.cosine\_similarity(  
    y\_true,  
    y\_pred,  
    axis=-1  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

tf.keras.losses.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/deserialize#aliases)

Aliases:

* tf.compat.v1.keras.losses.deserialize
* tf.compat.v2.keras.losses.deserialize
* tf.compat.v2.losses.deserialize
* tf.keras.losses.deserialize
* tf.losses.deserialize

tf.keras.losses.deserialize(  
    name,  
    custom\_objects=None  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

tf.keras.losses.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/get#aliases)

Aliases:

* tf.compat.v1.keras.losses.get
* tf.compat.v2.keras.losses.get
* tf.compat.v2.losses.get
* tf.keras.losses.get
* tf.losses.get

tf.keras.losses.get(identifier)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.Hinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Hinge#top_of_page)
* [Class Hinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Hinge#class_hinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Hinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Hinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Hinge#methods)

## Class Hinge

Computes the hinge loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.Hinge
* Class tf.compat.v2.keras.losses.Hinge
* Class tf.compat.v2.losses.Hinge
* Class tf.keras.losses.Hinge
* Class tf.losses.Hinge

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

y\_true values are expected to be -1 or 1. If binary (0 or 1) labels are provided we will convert them to -1 or 1.

#### Usage:

h = tf.keras.losses.Hinge()  
loss = h([-1., 1., 1.], [0.6, -0.7, -0.5])  
  
# loss = max(0, 1 - y\_true \* y\_pred) = [1.6 + 1.7 + 1.5] / 3  
  
print('Loss: ', loss.numpy())  # Loss: 1.6

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.Hinge())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name=None  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.hinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/hinge#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/hinge#aliases)

Computes the hinge loss between y\_true and y\_pred.

### Aliases:

* tf.compat.v1.keras.losses.hinge
* tf.compat.v1.keras.metrics.hinge
* tf.compat.v2.keras.losses.hinge
* tf.compat.v2.keras.metrics.hinge
* tf.compat.v2.losses.hinge
* tf.compat.v2.metrics.hinge
* tf.keras.losses.hinge
* tf.keras.metrics.hinge
* tf.losses.hinge
* tf.metrics.hinge

tf.keras.losses.hinge(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

#### Args:

* **y\_true**: The ground truth values. y\_true values are expected to be -1 or 1. If binary (0 or 1) labels are provided we will convert them to -1 or 1.
* **y\_pred**: The predicted values.

#### Returns:

Tensor with one scalar loss entry per sample.

# tf.keras.losses.Huber

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Huber#top_of_page)
* [Class Huber](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Huber#class_huber)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Huber#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Huber#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Huber#methods)

## Class Huber

Computes the Huber loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.Huber
* Class tf.compat.v2.keras.losses.Huber
* Class tf.compat.v2.losses.Huber
* Class tf.keras.losses.Huber
* Class tf.losses.Huber

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

For each value x in error=y\_true-y\_pred, the following is calculated:

0.5 \* x^2                  if |x| <= d  
0.5 \* d^2 + d \* (|x| - d)  if |x| > d

where d is delta. See: https://en.wikipedia.org/wiki/Huber\_loss

#### Usage:

l = tf.keras.losses.Huber()  
loss = l([0., 1., 1.], [1., 0., 1.])  
print('Loss: ', loss.numpy())  # Loss: 0.333

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.Huber())

#### Args:

* **delta**: A float, the point where the Huber loss function changes from a quadratic to linear.
* **reduction**: (Optional) Type of [tf.keras.losses.Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction) to apply to loss. Default value is AUTO. AUTO indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, using AUTO or SUM\_OVER\_BATCH\_SIZE will raise an error. Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.
* **name**: Optional name for the op.

## \_\_init\_\_

\_\_init\_\_(  
    delta=1.0,  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='huber\_loss'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.KLD

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD#aliases)

Aliases:

* tf.compat.v1.keras.losses.KLD
* tf.compat.v1.keras.losses.kld
* tf.compat.v1.keras.losses.kullback\_leibler\_divergence
* tf.compat.v1.keras.metrics.KLD
* tf.compat.v1.keras.metrics.kld
* tf.compat.v1.keras.metrics.kullback\_leibler\_divergence
* tf.compat.v2.keras.losses.KLD
* tf.compat.v2.keras.losses.kld
* tf.compat.v2.keras.losses.kullback\_leibler\_divergence
* tf.compat.v2.keras.metrics.KLD
* tf.compat.v2.keras.metrics.kld
* tf.compat.v2.keras.metrics.kullback\_leibler\_divergence
* tf.compat.v2.losses.KLD
* tf.compat.v2.losses.kld
* tf.compat.v2.losses.kullback\_leibler\_divergence
* tf.compat.v2.metrics.KLD
* tf.compat.v2.metrics.kld
* tf.compat.v2.metrics.kullback\_leibler\_divergence
* tf.keras.losses.KLD
* tf.keras.losses.kld
* tf.keras.losses.kullback\_leibler\_divergence
* tf.keras.metrics.KLD
* tf.keras.metrics.kld
* tf.keras.metrics.kullback\_leibler\_divergence
* tf.losses.KLD
* tf.losses.kld
* tf.losses.kullback\_leibler\_divergence
* tf.metrics.KLD
* tf.metrics.kld
* tf.metrics.kullback\_leibler\_divergence

tf.keras.losses.KLD(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.KLDivergence

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLDivergence#top_of_page)
* [Class KLDivergence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLDivergence#class_kldivergence)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLDivergence#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLDivergence#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLDivergence#methods)

## Class KLDivergence

Computes Kullback Leibler divergence loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.KLDivergence
* Class tf.compat.v2.keras.losses.KLDivergence
* Class tf.compat.v2.losses.KLDivergence
* Class tf.keras.losses.KLDivergence
* Class tf.losses.KLDivergence

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

loss = y\_true \* log(y\_true / y\_pred)

#### Usage:

k = tf.keras.losses.KLDivergence()  
loss = k([.4, .9, .2], [.5, .8, .12])  
print('Loss: ', loss.numpy())  # Loss: -0.043

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.KLDivergence())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='kullback\_leibler\_divergence'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.LogCosh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/LogCosh#top_of_page)
* [Class LogCosh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/LogCosh#class_logcosh)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/LogCosh#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/LogCosh#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/LogCosh#methods)

## Class LogCosh

Computes the logarithm of the hyperbolic cosine of the prediction error.

### Aliases:

* Class tf.compat.v1.keras.losses.LogCosh
* Class tf.compat.v2.keras.losses.LogCosh
* Class tf.compat.v2.losses.LogCosh
* Class tf.keras.losses.LogCosh
* Class tf.losses.LogCosh

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

logcosh = log((exp(x) + exp(-x))/2), where x is the error (y\_pred - y\_true)

#### Usage:

l = tf.keras.losses.LogCosh()  
loss = l([0., 1., 1.], [1., 0., 1.])  
print('Loss: ', loss.numpy())  # Loss: 0.289

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.LogCosh())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='logcosh'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.logcosh

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/logcosh#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/logcosh#aliases)

Logarithm of the hyperbolic cosine of the prediction error.

### Aliases:

* tf.compat.v1.keras.losses.logcosh
* tf.compat.v2.keras.losses.logcosh
* tf.compat.v2.losses.logcosh
* tf.keras.losses.logcosh
* tf.losses.logcosh

tf.keras.losses.logcosh(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

log(cosh(x)) is approximately equal to (x \*\* 2) / 2 for small x and to abs(x) - log(2) for large x. This means that 'logcosh' works mostly like the mean squared error, but will not be so strongly affected by the occasional wildly incorrect prediction.

#### Arguments:

* **y\_true**: tensor of true targets.
* **y\_pred**: tensor of predicted targets.

#### Returns:

Tensor with one scalar loss entry per sample.

# tf.keras.losses.Loss

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss#top_of_page)
* [Class Loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss#class_loss)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Loss#methods)

## Class Loss

Loss base class.

### Aliases:

* Class tf.compat.v1.keras.losses.Loss
* Class tf.compat.v2.keras.losses.Loss
* Class tf.compat.v2.losses.Loss
* Class tf.keras.losses.Loss
* Class tf.losses.Loss

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

To be implemented by subclasses: \* call(): Contains the logic for loss calculation using y\_true, y\_pred.

Example subclass implementation:

class MeanSquaredError(Loss):  
  def call(self, y\_true, y\_pred):  
    y\_pred = ops.convert\_to\_tensor(y\_pred)  
    y\_true = math\_ops.cast(y\_true, y\_pred.dtype)  
    return K.mean(math\_ops.square(y\_pred - y\_true), axis=-1)

When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras)compile and fit, please use 'SUM' or 'NONE' reduction types, and reduce losses explicitly in your training loop. Using 'AUTO' or 'SUM\_OVER\_BATCH\_SIZE' will raise an error.

Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.

You can implement 'SUM\_OVER\_BATCH\_SIZE' using global batch size like:

with strategy.scope():  
  loss\_obj = tf.keras.losses.CategoricalCrossentropy(  
      reduction=tf.keras.losses.Reduction.NONE)  
  ....  
  loss = (tf.reduce\_sum(loss\_obj(labels, predictions)) \*  
          (1. / global\_batch\_size))

#### Args:

* **reduction**: (Optional) Type of [tf.keras.losses.Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction) to apply to loss. Default value is AUTO. AUTO indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, using AUTO or SUM\_OVER\_BATCH\_SIZE will raise an error. Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.
* **name**: Optional name for the op.

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name=None  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### call

call(  
    y\_true,  
    y\_pred  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values, with the same shape as 'y\_pred'.
* **y\_pred**: The predicted values.

### from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.MAE

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE#aliases)

Aliases:

* tf.compat.v1.keras.losses.MAE
* tf.compat.v1.keras.losses.mae
* tf.compat.v1.keras.losses.mean\_absolute\_error
* tf.compat.v1.keras.metrics.MAE
* tf.compat.v1.keras.metrics.mae
* tf.compat.v1.keras.metrics.mean\_absolute\_error
* tf.compat.v2.keras.losses.MAE
* tf.compat.v2.keras.losses.mae
* tf.compat.v2.keras.losses.mean\_absolute\_error
* tf.compat.v2.keras.metrics.MAE
* tf.compat.v2.keras.metrics.mae
* tf.compat.v2.keras.metrics.mean\_absolute\_error
* tf.compat.v2.losses.MAE
* tf.compat.v2.losses.mae
* tf.compat.v2.losses.mean\_absolute\_error
* tf.compat.v2.metrics.MAE
* tf.compat.v2.metrics.mae
* tf.compat.v2.metrics.mean\_absolute\_error
* tf.keras.losses.MAE
* tf.keras.losses.mae
* tf.keras.losses.mean\_absolute\_error
* tf.keras.metrics.MAE
* tf.keras.metrics.mae
* tf.keras.metrics.mean\_absolute\_error
* tf.losses.MAE
* tf.losses.mae
* tf.losses.mean\_absolute\_error
* tf.metrics.MAE
* tf.metrics.mae
* tf.metrics.mean\_absolute\_error

tf.keras.losses.MAE(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

tf.keras.losses.MAPE

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE#aliases)

Aliases:

* tf.compat.v1.keras.losses.MAPE
* tf.compat.v1.keras.losses.mape
* tf.compat.v1.keras.losses.mean\_absolute\_percentage\_error
* tf.compat.v1.keras.metrics.MAPE
* tf.compat.v1.keras.metrics.mape
* tf.compat.v1.keras.metrics.mean\_absolute\_percentage\_error
* tf.compat.v2.keras.losses.MAPE
* tf.compat.v2.keras.losses.mape
* tf.compat.v2.keras.losses.mean\_absolute\_percentage\_error
* tf.compat.v2.keras.metrics.MAPE
* tf.compat.v2.keras.metrics.mape
* tf.compat.v2.keras.metrics.mean\_absolute\_percentage\_error
* tf.compat.v2.losses.MAPE
* tf.compat.v2.losses.mape
* tf.compat.v2.losses.mean\_absolute\_percentage\_error
* tf.compat.v2.metrics.MAPE
* tf.compat.v2.metrics.mape
* tf.compat.v2.metrics.mean\_absolute\_percentage\_error
* tf.keras.losses.MAPE
* tf.keras.losses.mape
* tf.keras.losses.mean\_absolute\_percentage\_error
* tf.keras.metrics.MAPE
* tf.keras.metrics.mape
* tf.keras.metrics.mean\_absolute\_percentage\_error
* tf.losses.MAPE
* tf.losses.mape
* tf.losses.mean\_absolute\_percentage\_error
* tf.metrics.MAPE
* tf.metrics.mape
* tf.metrics.mean\_absolute\_percentage\_error

tf.keras.losses.MAPE(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.MeanAbsoluteError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsoluteError#top_of_page)
* [Class MeanAbsoluteError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsoluteError#class_meanabsoluteerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsoluteError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsoluteError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsoluteError#methods)

## Class MeanAbsoluteError

Computes the mean of absolute difference between labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.losses.MeanAbsoluteError
* Class tf.compat.v2.keras.losses.MeanAbsoluteError
* Class tf.compat.v2.losses.MeanAbsoluteError
* Class tf.keras.losses.MeanAbsoluteError
* Class tf.losses.MeanAbsoluteError

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

For example, if y\_true is [0., 0., 1., 1.] and y\_pred is [1., 1., 1., 0.] then the mean absolute error value is 3/4 (0.75).

#### Usage:

mae = tf.keras.losses.MeanAbsoluteError()  
loss = mae([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Loss: ', loss.numpy())  # Loss: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.MeanAbsoluteError())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='mean\_absolute\_error'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.MeanAbsolutePercentageError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsolutePercentageError#top_of_page)
* [Class MeanAbsolutePercentageError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsolutePercentageError#class_meanabsolutepercentageerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsolutePercentageError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsolutePercentageError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanAbsolutePercentageError#methods)

## Class MeanAbsolutePercentageError

Computes the mean absolute percentage error between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.MeanAbsolutePercentageError
* Class tf.compat.v2.keras.losses.MeanAbsolutePercentageError
* Class tf.compat.v2.losses.MeanAbsolutePercentageError
* Class tf.keras.losses.MeanAbsolutePercentageError
* Class tf.losses.MeanAbsolutePercentageError

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

For example, if y\_true is [0., 0., 1., 1.] and y\_pred is [1., 1., 1., 0.] then the mean absolute percentage error value is 5e+08.

#### Usage:

mape = tf.keras.losses.MeanAbsolutePercentageError()  
loss = mape([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Loss: ', loss.numpy())  # Loss: 5e+08

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.MeanAbsolutePercentageError())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='mean\_absolute\_percentage\_error'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.MeanSquaredError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredError#top_of_page)
* [Class MeanSquaredError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredError#class_meansquarederror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredError#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredError#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredError#__init__)

## Class MeanSquaredError

Computes the mean of squares of errors between labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.losses.MeanSquaredError
* Class tf.compat.v2.keras.losses.MeanSquaredError
* Class tf.compat.v2.losses.MeanSquaredError
* Class tf.keras.losses.MeanSquaredError
* Class tf.losses.MeanSquaredError

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

For example, if y\_true is [0., 0., 1., 1.] and y\_pred is [1., 1., 1., 0.] then the mean squared error value is 3/4 (0.75).

#### Usage:

mse = tf.keras.losses.MeanSquaredError()  
loss = mse([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Loss: ', loss.numpy())  # Loss: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.MeanSquaredError())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='mean\_squared\_error'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.MeanSquaredLogarithmicError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError#top_of_page)
* [Class MeanSquaredLogarithmicError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError#class_meansquaredlogarithmicerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MeanSquaredLogarithmicError#methods)

## Class MeanSquaredLogarithmicError

Computes the mean squared logarithmic error between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.MeanSquaredLogarithmicError
* Class tf.compat.v2.keras.losses.MeanSquaredLogarithmicError
* Class tf.compat.v2.losses.MeanSquaredLogarithmicError
* Class tf.keras.losses.MeanSquaredLogarithmicError
* Class tf.losses.MeanSquaredLogarithmicError

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

For example, if y\_true is [0., 0., 1., 1.] and y\_pred is [1., 1., 1., 0.] then the mean squared logarithmic error value is 0.36034.

#### Usage:

msle = tf.keras.losses.MeanSquaredLogarithmicError()  
loss = msle([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Loss: ', loss.numpy())  # Loss: 0.36034

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.MeanSquaredLogarithmicError())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='mean\_squared\_logarithmic\_error'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.MSE

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE#aliases)

Aliases:

* tf.compat.v1.keras.losses.MSE
* tf.compat.v1.keras.losses.mean\_squared\_error
* tf.compat.v1.keras.losses.mse
* tf.compat.v1.keras.metrics.MSE
* tf.compat.v1.keras.metrics.mean\_squared\_error
* tf.compat.v1.keras.metrics.mse
* tf.compat.v2.keras.losses.MSE
* tf.compat.v2.keras.losses.mean\_squared\_error
* tf.compat.v2.keras.losses.mse
* tf.compat.v2.keras.metrics.MSE
* tf.compat.v2.keras.metrics.mean\_squared\_error
* tf.compat.v2.keras.metrics.mse
* tf.compat.v2.losses.MSE
* tf.compat.v2.losses.mean\_squared\_error
* tf.compat.v2.losses.mse
* tf.compat.v2.metrics.MSE
* tf.compat.v2.metrics.mean\_squared\_error
* tf.compat.v2.metrics.mse
* tf.keras.losses.MSE
* tf.keras.losses.mean\_squared\_error
* tf.keras.losses.mse
* tf.keras.metrics.MSE
* tf.keras.metrics.mean\_squared\_error
* tf.keras.metrics.mse
* tf.losses.MSE
* tf.losses.mean\_squared\_error
* tf.losses.mse
* tf.metrics.MSE
* tf.metrics.mean\_squared\_error
* tf.metrics.mse

tf.keras.losses.MSE(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

tf.keras.losses.MSLE

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE#aliases)

Aliases:

* tf.compat.v1.keras.losses.MSLE
* tf.compat.v1.keras.losses.mean\_squared\_logarithmic\_error
* tf.compat.v1.keras.losses.msle
* tf.compat.v1.keras.metrics.MSLE
* tf.compat.v1.keras.metrics.mean\_squared\_logarithmic\_error
* tf.compat.v1.keras.metrics.msle
* tf.compat.v2.keras.losses.MSLE
* tf.compat.v2.keras.losses.mean\_squared\_logarithmic\_error
* tf.compat.v2.keras.losses.msle
* tf.compat.v2.keras.metrics.MSLE
* tf.compat.v2.keras.metrics.mean\_squared\_logarithmic\_error
* tf.compat.v2.keras.metrics.msle
* tf.compat.v2.losses.MSLE
* tf.compat.v2.losses.mean\_squared\_logarithmic\_error
* tf.compat.v2.losses.msle
* tf.compat.v2.metrics.MSLE
* tf.compat.v2.metrics.mean\_squared\_logarithmic\_error
* tf.compat.v2.metrics.msle
* tf.keras.losses.MSLE
* tf.keras.losses.mean\_squared\_logarithmic\_error
* tf.keras.losses.msle
* tf.keras.metrics.MSLE
* tf.keras.metrics.mean\_squared\_logarithmic\_error
* tf.keras.metrics.msle
* tf.losses.MSLE
* tf.losses.mean\_squared\_logarithmic\_error
* tf.losses.msle
* tf.metrics.MSLE
* tf.metrics.mean\_squared\_logarithmic\_error
* tf.metrics.msle

tf.keras.losses.MSLE(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.Poisson

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Poisson#top_of_page)
* [Class Poisson](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Poisson#class_poisson)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Poisson#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Poisson#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Poisson#methods)

## Class Poisson

Computes the Poisson loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.Poisson
* Class tf.compat.v2.keras.losses.Poisson
* Class tf.compat.v2.losses.Poisson
* Class tf.keras.losses.Poisson
* Class tf.losses.Poisson

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

loss = y\_pred - y\_true \* log(y\_pred)

#### Usage:

p = tf.keras.losses.Poisson()  
loss = p([1, 9, 2], [4, 8, 12])  
print('Loss: ', loss.numpy())  # Loss: -4.63

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.Poisson())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='poisson'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.poisson

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/poisson#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/poisson#aliases)

Computes the Poisson loss between y\_true and y\_pred.

### Aliases:

* tf.compat.v1.keras.losses.poisson
* tf.compat.v1.keras.metrics.poisson
* tf.compat.v2.keras.losses.poisson
* tf.compat.v2.keras.metrics.poisson
* tf.compat.v2.losses.poisson
* tf.compat.v2.metrics.poisson
* tf.keras.losses.poisson
* tf.keras.metrics.poisson
* tf.losses.poisson
* tf.metrics.poisson

tf.keras.losses.poisson(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

The Poisson loss is the mean of the elements of the Tensor y\_pred - y\_true \* log(y\_pred).

#### Usage:

loss = tf.keras.losses.poisson([1.4, 9.3, 2.2], [4.3, 8.2, 12.2])  
print('Loss: ', loss.numpy())  # Loss: -0.8045559

#### Args:

* **y\_true**: Tensor of true targets.
* **y\_pred**: Tensor of predicted targets.

#### Returns:

A Tensor with the mean Poisson loss.

#### Raises:

* **InvalidArgumentError**: If y\_true and y\_pred have incompatible shapes.

tf.keras.losses.Reduction

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#top_of_page)
* [Class Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#class_reduction)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#methods)
  + [all](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#all)
  + [validate](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#validate)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#class_members)

Class Reduction

Types of loss reduction.

Aliases:

* Class tf.compat.v2.keras.losses.Reduction
* Class tf.compat.v2.losses.Reduction
* Class tf.keras.losses.Reduction
* Class tf.losses.Reduction

Defined in [python/ops/losses/loss\_reduction.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/losses/loss_reduction.py).

Contains the following values:

* AUTO: Indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, we expect reduction value to be SUM or NONE. Using AUTO in that case will raise an error.
* NONE: Un-reduced weighted losses with the same shape as input. When this reduction type used with built-in Keras training loops like fit/evaluate, the unreduced vector loss is passed to the optimizer but the reported loss will be a scalar value.
* SUM: Scalar sum of weighted losses.
* SUM\_OVER\_BATCH\_SIZE: Scalar SUM divided by number of elements in losses. This reduction type is not supported when used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy) outside of built-in training loops like [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile/fit.

You can implement 'SUM\_OVER\_BATCH\_SIZE' using global batch size like:

with strategy.scope():  
  loss\_obj = tf.keras.losses.CategoricalCrossentropy(  
      reduction=tf.keras.losses.Reduction.None)  
  ....  
  loss = tf.reduce\_sum(loss\_object(labels, predictions)) \*  
      (1. / global\_batch\_size)

Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.

Methods

all

@classmethod  
all(cls)

validate

@classmethod  
validate(  
    cls,  
    key  
)

Class Members

* AUTO = 'auto'
* NONE = 'none'
* SUM = 'sum'
* SUM\_OVER\_BATCH\_SIZE = 'sum\_over\_batch\_size'

tf.keras.losses.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/serialize#aliases)

Aliases:

* tf.compat.v1.keras.losses.serialize
* tf.compat.v2.keras.losses.serialize
* tf.compat.v2.losses.serialize
* tf.keras.losses.serialize
* tf.losses.serialize

tf.keras.losses.serialize(loss)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

# tf.keras.losses.SparseCategoricalCrossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy#top_of_page)
* [Class SparseCategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy#class_sparsecategoricalcrossentropy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy#used_in_the_tutorials)

## Class SparseCategoricalCrossentropy

Computes the crossentropy loss between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.losses.SparseCategoricalCrossentropy
* Class tf.compat.v2.keras.losses.SparseCategoricalCrossentropy
* Class tf.compat.v2.losses.SparseCategoricalCrossentropy
* Class tf.keras.losses.SparseCategoricalCrossentropy
* Class tf.losses.SparseCategoricalCrossentropy

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

Use this crossentropy loss function when there are two or more label classes. We expect labels to be provided as integers. If you want to provide labels using one-hot representation, please use CategoricalCrossentropy loss. There should be # classes floating point values per feature for y\_pred and a single floating point value per feature for y\_true.

In the snippet below, there is a single floating point value per example for y\_true and # classesfloating pointing values per example for y\_pred. The shape of y\_true is [batch\_size] and the shape of y\_pred is [batch\_size, num\_classes].

#### Usage:

cce = tf.keras.losses.SparseCategoricalCrossentropy()  
loss = cce(  
  [0, 1, 2],  
  [[.9, .05, .05], [.5, .89, .6], [.05, .01, .94]])  
print('Loss: ', loss.numpy())  # Loss: 0.3239

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.SparseCategoricalCrossentropy())

#### Args:

* **from\_logits**: Whether y\_pred is expected to be a logits tensor. By default, we assume that y\_pred encodes a probability distribution.
* **reduction**: (Optional) Type of [tf.keras.losses.Reduction](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction) to apply to loss. Default value is AUTO. AUTO indicates that the reduction option will be determined by the usage context. For almost all cases this defaults to SUM\_OVER\_BATCH\_SIZE. When used with [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), outside of built-in training loops such as [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) compile and fit, using AUTO or SUM\_OVER\_BATCH\_SIZE will raise an error. Please see https://www.tensorflow.org/alpha/tutorials/distribute/training\_loops for more details on this.
* **name**: Optional name for the op.

## \_\_init\_\_

\_\_init\_\_(  
    from\_logits=False,  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name=None  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

tf.keras.losses.sparse\_categorical\_crossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/sparse_categorical_crossentropy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/sparse_categorical_crossentropy#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/sparse_categorical_crossentropy#used_in_the_tutorials)

Aliases:

* tf.compat.v1.keras.losses.sparse\_categorical\_crossentropy
* tf.compat.v1.keras.metrics.sparse\_categorical\_crossentropy
* tf.compat.v2.keras.losses.sparse\_categorical\_crossentropy
* tf.compat.v2.keras.metrics.sparse\_categorical\_crossentropy
* tf.compat.v2.losses.sparse\_categorical\_crossentropy
* tf.compat.v2.metrics.sparse\_categorical\_crossentropy
* tf.keras.losses.sparse\_categorical\_crossentropy
* tf.keras.metrics.sparse\_categorical\_crossentropy
* tf.losses.sparse\_categorical\_crossentropy
* tf.metrics.sparse\_categorical\_crossentropy

tf.keras.losses.sparse\_categorical\_crossentropy(  
    y\_true,  
    y\_pred,  
    from\_logits=False,  
    axis=-1  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

Used in the tutorials:

* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)

# tf.keras.losses.SquaredHinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SquaredHinge#top_of_page)
* [Class SquaredHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SquaredHinge#class_squaredhinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SquaredHinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SquaredHinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/SquaredHinge#methods)

## Class SquaredHinge

Computes the squared hinge loss between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.losses.SquaredHinge
* Class tf.compat.v2.keras.losses.SquaredHinge
* Class tf.compat.v2.losses.SquaredHinge
* Class tf.keras.losses.SquaredHinge
* Class tf.losses.SquaredHinge

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

y\_true values are expected to be -1 or 1. If binary (0 or 1) labels are provided we will convert them to -1 or 1.

#### Usage:

sh = tf.keras.losses.SquaredHinge()  
loss = sh([-1., 1., 1.], [0.6, -0.7, -0.5])  
  
# loss = (max(0, 1 - y\_true \* y\_pred))^2 = [1.6^2 + 1.7^2 + 1.5^2] / 3  
  
print('Loss: ', loss.numpy())  # Loss: 2.566666

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss=tf.keras.losses.SquaredHinge())

## \_\_init\_\_

\_\_init\_\_(  
    reduction=losses\_utils.ReductionV2.AUTO,  
    name='squared\_hinge'  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Invokes the Loss instance.

#### Args:

* **y\_true**: Ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional Tensor whose rank is either 0, or the same rank as y\_true, or is broadcastable to y\_true. sample\_weight acts as a coefficient for the loss. If a scalar is provided, then the loss is simply scaled by the given value. If sample\_weight is a tensor of size[batch\_size], then the total loss for each sample of the batch is rescaled by the corresponding element in the sample\_weight vector. If the shape of sample\_weight matches the shape of y\_pred, then the loss of each measurable element of y\_pred is scaled by the corresponding value of sample\_weight.

#### Returns:

Weighted loss float Tensor. If reduction is NONE, this has the same shape as y\_true; otherwise, it is scalar.

#### Raises:

* **ValueError**: If the shape of sample\_weight is invalid.

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a Loss from its config (output of get\_config()).

#### Args:

* **config**: Output of get\_config().

#### Returns:

A Loss instance.

### get\_config

get\_config()

# tf.keras.losses.squared\_hinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/squared_hinge#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/squared_hinge#aliases)

Computes the squared hinge loss between y\_true and y\_pred.

### Aliases:

* tf.compat.v1.keras.losses.squared\_hinge
* tf.compat.v1.keras.metrics.squared\_hinge
* tf.compat.v2.keras.losses.squared\_hinge
* tf.compat.v2.keras.metrics.squared\_hinge
* tf.compat.v2.losses.squared\_hinge
* tf.compat.v2.metrics.squared\_hinge
* tf.keras.losses.squared\_hinge
* tf.keras.metrics.squared\_hinge
* tf.losses.squared\_hinge
* tf.metrics.squared\_hinge

tf.keras.losses.squared\_hinge(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/losses.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/losses.py).

#### Args:

* **y\_true**: The ground truth values. y\_true values are expected to be -1 or 1. If binary (0 or 1) labels are provided we will convert them to -1 or 1.
* **y\_pred**: The predicted values.

#### Returns:

Tensor with one scalar loss entry per sample.

Module: tf.compat.v1.metrics

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics#top_of_page)

[Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics#functions)

Evaluation-related metrics.

Functions

[accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/accuracy): Calculates how often predictions matches labels.

[auc(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/auc): Computes the approximate AUC via a Riemann sum.

[average\_precision\_at\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/average_precision_at_k): Computes average precision@k of predictions with respect to sparse labels.

[false\_negatives(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/false_negatives): Computes the total number of false negatives.

[false\_negatives\_at\_thresholds(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/false_negatives_at_thresholds): Computes false negatives at provided threshold values.

[false\_positives(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/false_positives): Sum the weights of false positives.

[false\_positives\_at\_thresholds(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/false_positives_at_thresholds): Computes false positives at provided threshold values.

[mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean): Computes the (weighted) mean of the given values.

[mean\_absolute\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_absolute_error): Computes the mean absolute error between the labels and predictions.

[mean\_cosine\_distance(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_cosine_distance): Computes the cosine distance between the labels and predictions.

[mean\_iou(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_iou): Calculate per-step mean Intersection-Over-Union (mIOU).

[mean\_per\_class\_accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_per_class_accuracy): Calculates the mean of the per-class accuracies.

[mean\_relative\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_relative_error): Computes the mean relative error by normalizing with the given values.

[mean\_squared\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_squared_error): Computes the mean squared error between the labels and predictions.

[mean\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/mean_tensor): Computes the element-wise (weighted) mean of the given tensors.

[percentage\_below(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/percentage_below): Computes the percentage of values less than the given threshold.

[precision(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/precision): Computes the precision of the predictions with respect to the labels.

[precision\_at\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/precision_at_k): Computes precision@k of the predictions with respect to sparse labels.

[precision\_at\_thresholds(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/precision_at_thresholds): Computes precision values for different thresholds on predictions.

[precision\_at\_top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/precision_at_top_k): Computes precision@k of the predictions with respect to sparse labels.

[recall(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/recall): Computes the recall of the predictions with respect to the labels.

[recall\_at\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/recall_at_k): Computes recall@k of the predictions with respect to sparse labels.

[recall\_at\_thresholds(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/recall_at_thresholds): Computes various recall values for different thresholds on predictions.

[recall\_at\_top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/recall_at_top_k): Computes recall@k of top-k predictions with respect to sparse labels.

[root\_mean\_squared\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/root_mean_squared_error): Computes the root mean squared error between the labels and predictions.

[sensitivity\_at\_specificity(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/sensitivity_at_specificity): Computes the specificity at a given sensitivity.

[sparse\_average\_precision\_at\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/sparse_average_precision_at_k): Renamed to average\_precision\_at\_k, please use that method instead. (deprecated)

[sparse\_precision\_at\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/sparse_precision_at_k): Renamed to precision\_at\_k, please use that method instead. (deprecated)

[specificity\_at\_sensitivity(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/specificity_at_sensitivity): Computes the specificity at a given sensitivity.

[true\_negatives(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/true_negatives): Sum the weights of true\_negatives.

[true\_negatives\_at\_thresholds(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/true_negatives_at_thresholds): Computes true negatives at provided threshold values.

[true\_positives(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/true_positives): Sum the weights of true\_positives.

[true\_positives\_at\_thresholds(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/metrics/true_positives_at_thresholds): Computes true positives at provided threshold values.

Module: tf.keras.metrics

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics#functions)

Built-in metrics.

Aliases:

* Module tf.compat.v2.keras.metrics
* Module tf.compat.v2.metrics
* Module tf.keras.metrics
* Module tf.metrics

Defined in [python/keras/api/\_v2/keras/metrics/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/metrics/__init__.py).

Classes

[class AUC](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/AUC): Computes the approximate AUC (Area under the curve) via a Riemann sum.

[class Accuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Accuracy): Calculates how often predictions matches labels.

[class BinaryAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryAccuracy): Calculates how often predictions matches labels.

[class BinaryCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryCrossentropy): Computes the crossentropy metric between the labels and predictions.

[class CategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalAccuracy): Calculates how often predictions matches labels.

[class CategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalCrossentropy): Computes the crossentropy metric between the labels and predictions.

[class CategoricalHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalHinge): Computes the categorical hinge metric between y\_true and y\_pred.

[class CosineSimilarity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CosineSimilarity): Computes the cosine similarity between the labels and predictions.

[class FalseNegatives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalseNegatives): Calculates the number of false negatives.

[class FalsePositives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalsePositives): Calculates the number of false positives.

[class Hinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Hinge): Computes the hinge metric between y\_true and y\_pred.

[class KLDivergence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/KLDivergence): Computes Kullback-Leibler divergence metric between y\_true and y\_pred.

[class LogCoshError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/LogCoshError): Computes the logarithm of the hyperbolic cosine of the prediction error.

[class Mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean): Computes the (weighted) mean of the given values.

[class MeanAbsoluteError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsoluteError): Computes the mean absolute error between the labels and predictions.

[class MeanAbsolutePercentageError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsolutePercentageError): Computes the mean absolute percentage error between y\_true and y\_pred.

[class MeanIoU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanIoU): Computes the mean Intersection-Over-Union metric.

[class MeanRelativeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanRelativeError): Computes the mean relative error by normalizing with the given values.

[class MeanSquaredError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredError): Computes the mean squared error between y\_true and y\_pred.

[class MeanSquaredLogarithmicError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredLogarithmicError): Computes the mean squared logarithmic error between y\_true and y\_pred.

[class MeanTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanTensor): Computes the element-wise (weighted) mean of the given tensors.

[class Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric): Encapsulates metric logic and state.

[class Poisson](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Poisson): Computes the Poisson metric between y\_true and y\_pred.

[class Precision](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Precision): Computes the precision of the predictions with respect to the labels.

[class Recall](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Recall): Computes the recall of the predictions with respect to the labels.

[class RootMeanSquaredError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/RootMeanSquaredError): Computes root mean squared error metric between y\_true and y\_pred.

[class SensitivityAtSpecificity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity): Computes the sensitivity at a given specificity.

[class SparseCategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy): Calculates how often predictions matches integer labels.

[class SparseCategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalCrossentropy): Computes the crossentropy metric between the labels and predictions.

[class SparseTopKCategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseTopKCategoricalAccuracy): Computes how often integer targets are in the top Kpredictions.

[class SpecificityAtSensitivity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SpecificityAtSensitivity): Computes the specificity at a given sensitivity.

[class SquaredHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SquaredHinge): Computes the squared hinge metric between y\_true and y\_pred.

[class Sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Sum): Computes the (weighted) sum of the given values.

[class TopKCategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TopKCategoricalAccuracy): Computes how often targets are in the top K predictions.

[class TrueNegatives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TrueNegatives): Calculates the number of true negatives.

[class TruePositives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TruePositives): Calculates the number of true positives.

Functions

[KLD(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD)

[MAE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE)

[MAPE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE)

[MSE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE)

[MSLE(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE)

[binary\_accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/binary_accuracy)

[binary\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/binary_crossentropy)

[categorical\_accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/categorical_accuracy)

[categorical\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/categorical_crossentropy): Computes the categorical crossentropy loss.

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/deserialize)

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/get)

[hinge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/hinge): Computes the hinge loss between y\_true and y\_pred.

[kld(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD)

[kullback\_leibler\_divergence(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/KLD)

[mae(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE)

[mape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE)

[mean\_absolute\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAE)

[mean\_absolute\_percentage\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MAPE)

[mean\_squared\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE)

[mean\_squared\_logarithmic\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE)

[mse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSE)

[msle(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/MSLE)

[poisson(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/poisson): Computes the Poisson loss between y\_true and y\_pred.

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/serialize)

[sparse\_categorical\_accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/sparse_categorical_accuracy)

[sparse\_categorical\_crossentropy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/sparse_categorical_crossentropy)

[sparse\_top\_k\_categorical\_accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/sparse_top_k_categorical_accuracy)

[squared\_hinge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/squared_hinge): Computes the squared hinge loss between y\_true and y\_pred.

[top\_k\_categorical\_accuracy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/top_k_categorical_accuracy)

# tf.keras.metrics.Accuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Accuracy#top_of_page)
* [Class Accuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Accuracy#class_accuracy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Accuracy#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Accuracy#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Accuracy#__init__)

## Class Accuracy

Calculates how often predictions matches labels.

### Aliases:

* Class tf.compat.v1.keras.metrics.Accuracy
* Class tf.compat.v2.keras.metrics.Accuracy
* Class tf.compat.v2.metrics.Accuracy
* Class tf.keras.metrics.Accuracy
* Class tf.metrics.Accuracy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)

For example, if y\_true is [1, 2, 3, 4] and y\_pred is [0, 2, 3, 4] then the accuracy is 3/4 or .75. If the weights were specified as [1, 1, 0, 0] then the accuracy would be 1/2 or .5.

This metric creates two local variables, total and count that are used to compute the frequency with which y\_pred matches y\_true. This frequency is ultimately returned as binary accuracy: an idempotent operation that simply divides total by count.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.Accuracy()  
m.update\_state([1, 2, 3, 4], [0, 2, 3, 4])  
print('Final result: ', m.result().numpy())  # Final result: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.Accuracy()])

## \_\_init\_\_

\_\_init\_\_(  
    name='accuracy',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.AUC

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/AUC#top_of_page)
* [Class AUC](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/AUC#class_auc)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/AUC#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/AUC#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/AUC#methods)

## Class AUC

Computes the approximate AUC (Area under the curve) via a Riemann sum.

Inherits From: [Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric)

### Aliases:

* Class tf.compat.v1.keras.metrics.AUC
* Class tf.compat.v2.keras.metrics.AUC
* Class tf.compat.v2.metrics.AUC
* Class tf.keras.metrics.AUC
* Class tf.metrics.AUC

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

This metric creates four local variables, true\_positives, true\_negatives, false\_positives and false\_negatives that are used to compute the AUC. To discretize the AUC curve, a linearly spaced set of thresholds is used to compute pairs of recall and precision values. The area under the ROC-curve is therefore computed using the height of the recall values by the false positive rate, while the area under the PR-curve is the computed using the height of the precision values by the recall.

This value is ultimately returned as auc, an idempotent operation that computes the area under a discretized curve of precision versus recall values (computed using the aforementioned variables). The num\_thresholds variable controls the degree of discretization with larger numbers of thresholds more closely approximating the true AUC. The quality of the approximation may vary dramatically depending on num\_thresholds. The thresholds parameter can be used to manually specify thresholds which split the predictions more evenly.

For best results, predictions should be distributed approximately uniformly in the range [0, 1] and not peaked around 0 or 1. The quality of the AUC approximation may be poor if this is not the case. Setting summation\_method to 'minoring' or 'majoring' can help quantify the error in the approximation by providing lower or upper bound estimate of the AUC.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.AUC(num\_thresholds=3)  
m.update\_state([0, 0, 1, 1], [0, 0.5, 0.3, 0.9])  
  
# threshold values are [0 - 1e-7, 0.5, 1 + 1e-7]  
# tp = [2, 1, 0], fp = [2, 0, 0], fn = [0, 1, 2], tn = [0, 2, 2]  
# recall = [1, 0.5, 0], fp\_rate = [1, 0, 0]  
# auc = ((((1+0.5)/2)\*(1-0))+ (((0.5+0)/2)\*(0-0))) = 0.75  
  
print('Final result: ', m.result().numpy())  # Final result: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.AUC()])

## \_\_init\_\_

\_\_init\_\_(  
    num\_thresholds=200,  
    curve='ROC',  
    summation\_method='interpolation',  
    name=None,  
    dtype=None,  
    thresholds=None  
)

Creates an AUC instance.

#### Args:

* **num\_thresholds**: (Optional) Defaults to 200. The number of thresholds to use when discretizing the roc curve. Values must be > 1.
* **curve**: (Optional) Specifies the name of the curve to be computed, 'ROC' [default] or 'PR' for the Precision-Recall-curve.
* **summation\_method**: (Optional) Specifies the Riemann summation method used (https://en.wikipedia.org/wiki/Riemann\_sum): 'interpolation' [default], applies mid-point summation scheme for ROC. For PR-AUC, interpolates (true/false) positives but not the ratio that is precision (see Davis & Goadrich 2006 for details); 'minoring' that applies left summation for increasing intervals and right summation for decreasing intervals; 'majoring' that does the opposite.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.
* **thresholds**: (Optional) A list of floating point values to use as the thresholds for discretizing the curve. If set, the num\_thresholds parameter is ignored. Values should be in [0, 1]. Endpoint thresholds equal to {-epsilon, 1+epsilon} for a small positive epsilon value will be automatically included with these to correctly handle predictions equal to exactly 0 or 1.

## Methods

### interpolate\_pr\_auc

interpolate\_pr\_auc()

Interpolation formula inspired by section 4 of Davis & Goadrich 2006.

https://www.biostat.wisc.edu/~page/rocpr.pdf

Note here we derive & use a closed formula not present in the paper as follows:

Precision = TP / (TP + FP) = TP / P

Modeling all of TP (true positive), FP (false positive) and their sum P = TP + FP (predicted positive) as varying linearly within each interval [A, B] between successive thresholds, we get

Precision slope = dTP / dP = (TP\_B - TP\_A) / (P\_B - P\_A) = (TP - TP\_A) / (P - P\_A) Precision = (TP\_A + slope \* (P - P\_A)) / P

The area within the interval is (slope / total\_pos\_weight) times

int\_A^B{Precision.dP} = int\_A^B{(TP\_A + slope \* (P - P\_A)) \* dP / P} int\_A^B{Precision.dP} = int\_A^B{slope \* dP + intercept \* dP / P}

where intercept = TP\_A - slope \* P\_A = TP\_B - slope \* P\_B, resulting in

int\_A^B{Precision.dP} = TP\_B - TP\_A + intercept \* log(P\_B / P\_A)

Bringing back the factor (slope / total\_pos\_weight) we'd put aside, we get

slope \* [dTP + intercept \* log(P\_B / P\_A)] / total\_pos\_weight

where dTP == TP\_B - TP\_A.

Note that when P\_A == 0 the above calculation simplifies into

int\_A^B{Precision.dTP} = int\_A^B{slope \* dTP} = slope \* (TP\_B - TP\_A)

which is really equivalent to imputing constant precision throughout the first bucket having >0 true positives.

#### Returns:

* **pr\_auc**: an approximation of the area under the P-R curve.

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates confusion matrix statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.BinaryAccuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryAccuracy#top_of_page)
* [Class BinaryAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryAccuracy#class_binaryaccuracy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryAccuracy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryAccuracy#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryAccuracy#methods)

## Class BinaryAccuracy

Calculates how often predictions matches labels.

### Aliases:

* Class tf.compat.v1.keras.metrics.BinaryAccuracy
* Class tf.compat.v2.keras.metrics.BinaryAccuracy
* Class tf.compat.v2.metrics.BinaryAccuracy
* Class tf.keras.metrics.BinaryAccuracy
* Class tf.metrics.BinaryAccuracy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [1, 1, 0, 0] and y\_pred is [0.98, 1, 0, 0.6] then the binary accuracy is 3/4 or .75. If the weights were specified as [1, 0, 0, 1] then the binary accuracy would be 1/2 or .5.

This metric creates two local variables, total and count that are used to compute the frequency with which y\_pred matches y\_true. This frequency is ultimately returned as binary accuracy: an idempotent operation that simply divides total by count.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.BinaryAccuracy()  
m.update\_state([1, 1, 0, 0], [0.98, 1, 0, 0.6])  
print('Final result: ', m.result().numpy())  # Final result: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.BinaryAccuracy()])

## \_\_init\_\_

\_\_init\_\_(  
    name='binary\_accuracy',  
    dtype=None,  
    threshold=0.5  
)

Creates a BinaryAccuracy instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.
* **threshold**: (Optional) Float representing the threshold for deciding whether prediction values are 1 or 0.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.BinaryCrossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryCrossentropy#top_of_page)
* [Class BinaryCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryCrossentropy#class_binarycrossentropy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryCrossentropy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryCrossentropy#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/BinaryCrossentropy#methods)

## Class BinaryCrossentropy

Computes the crossentropy metric between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.BinaryCrossentropy
* Class tf.compat.v2.keras.metrics.BinaryCrossentropy
* Class tf.compat.v2.metrics.BinaryCrossentropy
* Class tf.keras.metrics.BinaryCrossentropy
* Class tf.metrics.BinaryCrossentropy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

This is the crossentropy metric class to be used when there are only two label classes (0 and 1).

#### Usage:

m = tf.keras.metrics.BinaryCrossentropy()  
m.update\_state([1., 0., 1., 0.], [1., 1., 1., 0.])  
  
# EPSILON = 1e-7, y = y\_true, y` = y\_pred, Y\_MAX = 0.9999999  
# y` = clip\_ops.clip\_by\_value(output, EPSILON, 1. - EPSILON)  
# y` = [Y\_MAX, Y\_MAX, Y\_MAX, EPSILON]  
  
# Metric = -(y log(y` + EPSILON) + (1 - y) log(1 - y` + EPSILON))  
#        = [-log(Y\_MAX + EPSILON), -log(1 - Y\_MAX + EPSILON),  
#           -log(Y\_MAX + EPSILON), -log(1)]  
#        = [(0 + 15.33) / 2, (0 + 0) / 2]  
# Reduced metric = 7.665 / 2  
  
print('Final result: ', m.result().numpy())  # Final result: 3.833

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
    'sgd',  
    loss='mse',  
    metrics=[tf.keras.metrics.BinaryCrossentropy()])

## \_\_init\_\_

\_\_init\_\_(  
    name='binary\_crossentropy',  
    dtype=None,  
    from\_logits=False,  
    label\_smoothing=0  
)

Creates a BinaryCrossentropy instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.
* **from\_logits**: (Optional )Whether output is expected to be a logits tensor. By default, we consider that output encodes a probability distribution.
* **label\_smoothing**: (Optional) Float in [0, 1]. When > 0, label values are smoothed, meaning the confidence on label values are relaxed. e.g. label\_smoothing=0.2 means that we will use a value of 0.1 for label 0 and 0.9 for label 1"

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.binary\_accuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/binary_accuracy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/binary_accuracy#aliases)

Aliases:

* tf.compat.v1.keras.metrics.binary\_accuracy
* tf.compat.v2.keras.metrics.binary\_accuracy
* tf.compat.v2.metrics.binary\_accuracy
* tf.keras.metrics.binary\_accuracy
* tf.metrics.binary\_accuracy

tf.keras.metrics.binary\_accuracy(  
    y\_true,  
    y\_pred,  
    threshold=0.5  
)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

# tf.keras.metrics.CategoricalAccuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalAccuracy#top_of_page)
* [Class CategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalAccuracy#class_categoricalaccuracy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalAccuracy#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalAccuracy#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalAccuracy#__init__)

## Class CategoricalAccuracy

Calculates how often predictions matches labels.

### Aliases:

* Class tf.compat.v1.keras.metrics.CategoricalAccuracy
* Class tf.compat.v2.keras.metrics.CategoricalAccuracy
* Class tf.compat.v2.metrics.CategoricalAccuracy
* Class tf.keras.metrics.CategoricalAccuracy
* Class tf.metrics.CategoricalAccuracy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

For example, if y\_true is [[0, 0, 1], [0, 1, 0]] and y\_pred is [[0.1, 0.9, 0.8], [0.05, 0.95, 0]] then the categorical accuracy is 1/2 or .5. If the weights were specified as [0.7, 0.3] then the categorical accuracy would be .3. You can provide logits of classes as y\_pred, since argmax of logits and probabilities are same.

This metric creates two local variables, total and count that are used to compute the frequency with which y\_pred matches y\_true. This frequency is ultimately returned as categorical accuracy: an idempotent operation that simply divides total by count.

y\_pred and y\_true should be passed in as vectors of probabilities, rather than as labels. If necessary, use [tf.one\_hot](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/one_hot) to expand y\_true as a vector.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.CategoricalAccuracy()  
m.update\_state([[0, 0, 1], [0, 1, 0]], [[0.1, 0.9, 0.8], [0.05, 0.95, 0]])  
print('Final result: ', m.result().numpy())  # Final result: 0.5

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
  'sgd',  
  loss='mse',  
  metrics=[tf.keras.metrics.CategoricalAccuracy()])

## \_\_init\_\_

\_\_init\_\_(  
    name='categorical\_accuracy',  
    dtype=None  
)

Creates a CategoricalAccuracy instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.CategoricalCrossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalCrossentropy#top_of_page)
* [Class CategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalCrossentropy#class_categoricalcrossentropy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalCrossentropy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalCrossentropy#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalCrossentropy#methods)

## Class CategoricalCrossentropy

Computes the crossentropy metric between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.CategoricalCrossentropy
* Class tf.compat.v2.keras.metrics.CategoricalCrossentropy
* Class tf.compat.v2.metrics.CategoricalCrossentropy
* Class tf.keras.metrics.CategoricalCrossentropy
* Class tf.metrics.CategoricalCrossentropy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

This is the crossentropy metric class to be used when there are multiple label classes (2 or more). Here we assume that labels are given as a one\_hot representation. eg., When labels values are [2, 0, 1], y\_true = [[0, 0, 1], [1, 0, 0], [0, 1, 0]].

#### Usage:

m = tf.keras.metrics.CategoricalCrossentropy()  
m.update\_state([[0, 1, 0], [0, 0, 1]],  
               [[0.05, 0.95, 0], [0.1, 0.8, 0.1]])  
  
# EPSILON = 1e-7, y = y\_true, y` = y\_pred  
# y` = clip\_ops.clip\_by\_value(output, EPSILON, 1. - EPSILON)  
# y` = [[0.05, 0.95, EPSILON], [0.1, 0.8, 0.1]]  
  
# xent = -sum(y \* log(y'), axis = -1)  
#      = -((log 0.95), (log 0.1))  
#      = [0.051, 2.302]  
# Reduced xent = (0.051 + 2.302) / 2  
  
print('Final result: ', m.result().numpy())  # Final result: 1.176

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
  'sgd',  
  loss='mse',  
  metrics=[tf.keras.metrics.CategoricalCrossentropy()])

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.
* **from\_logits**: (Optional ) Whether y\_pred is expected to be a logits tensor. By default, we assume that y\_pred encodes a probability distribution.
* **label\_smoothing**: Float in [0, 1]. When > 0, label values are smoothed, meaning the confidence on label values are relaxed. e.g. label\_smoothing=0.2 means that we will use a value of 0.1for label 0 and 0.9 for label 1"

## \_\_init\_\_

\_\_init\_\_(  
    name='categorical\_crossentropy',  
    dtype=None,  
    from\_logits=False,  
    label\_smoothing=0  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.CategoricalHinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalHinge#top_of_page)
* [Class CategoricalHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalHinge#class_categoricalhinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalHinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalHinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CategoricalHinge#methods)

## Class CategoricalHinge

Computes the categorical hinge metric between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.CategoricalHinge
* Class tf.compat.v2.keras.metrics.CategoricalHinge
* Class tf.compat.v2.metrics.CategoricalHinge
* Class tf.keras.metrics.CategoricalHinge
* Class tf.metrics.CategoricalHinge

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0., 1., 1.], and y\_pred is [1., 0., 1.] the categorical hinge metric value is 1.0.

#### Usage:

m = tf.keras.metrics.CategoricalHinge()  
m.update\_state([0., 1., 1.], [1., 0., 1.])  
print('Final result: ', m.result().numpy())  # Final result: 1.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.CategoricalHinge()])

## \_\_init\_\_

\_\_init\_\_(  
    name='categorical\_hinge',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.categorical\_accuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/categorical_accuracy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/categorical_accuracy#aliases)

Aliases:

* tf.compat.v1.keras.metrics.categorical\_accuracy
* tf.compat.v2.keras.metrics.categorical\_accuracy
* tf.compat.v2.metrics.categorical\_accuracy
* tf.keras.metrics.categorical\_accuracy
* tf.metrics.categorical\_accuracy

tf.keras.metrics.categorical\_accuracy(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py)

# tf.keras.metrics.CosineSimilarity

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CosineSimilarity#top_of_page)
* [Class CosineSimilarity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CosineSimilarity#class_cosinesimilarity)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CosineSimilarity#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CosineSimilarity#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/CosineSimilarity#methods)

## Class CosineSimilarity

Computes the cosine similarity between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.CosineSimilarity
* Class tf.compat.v2.keras.metrics.CosineSimilarity
* Class tf.compat.v2.metrics.CosineSimilarity
* Class tf.keras.metrics.CosineSimilarity
* Class tf.metrics.CosineSimilarity

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

cosine similarity = (a . b) / ||a|| ||b|| [Cosine Similarity](https://en.wikipedia.org/wiki/Cosine_similarity)

For example, if y\_true is [0, 1, 1], and y\_pred is [1, 0, 1], the cosine similarity is 0.5.

This metric keeps the average cosine similarity between predictions and labels over a stream of data.

#### Usage:

m = tf.keras.metrics.CosineSimilarity(axis=1)  
m.update\_state([[0., 1.], [1., 1.]], [[1., 0.], [1., 1.]])  
# l2\_norm(y\_true) = [[0., 1.], [1./1.414], 1./1.414]]]  
# l2\_norm(y\_pred) = [[1., 0.], [1./1.414], 1./1.414]]]  
# l2\_norm(y\_true) . l2\_norm(y\_pred) = [[0., 0.], [0.5, 0.5]]  
# result = mean(sum(l2\_norm(y\_true) . l2\_norm(y\_pred), axis=1))  
       = ((0. + 0.) +  (0.5 + 0.5)) / 2  
  
print('Final result: ', m.result().numpy())  # Final result: 0.5

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
    'sgd',  
    loss='mse',  
    metrics=[tf.keras.metrics.CosineSimilarity(axis=1)])

## \_\_init\_\_

\_\_init\_\_(  
    name='cosine\_similarity',  
    dtype=None,  
    axis=-1  
)

Creates a CosineSimilarity instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.
* **axis**: (Optional) Defaults to -1. The dimension along which the cosine similarity is computed.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/deserialize#aliases)

Aliases:

* tf.compat.v1.keras.metrics.deserialize
* tf.compat.v2.keras.metrics.deserialize
* tf.compat.v2.metrics.deserialize
* tf.keras.metrics.deserialize
* tf.metrics.deserialize

tf.keras.metrics.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

# tf.keras.metrics.FalseNegatives

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalseNegatives#top_of_page)
* [Class FalseNegatives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalseNegatives#class_falsenegatives)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalseNegatives#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalseNegatives#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalseNegatives#methods)

## Class FalseNegatives

Calculates the number of false negatives.

### Aliases:

* Class tf.compat.v1.keras.metrics.FalseNegatives
* Class tf.compat.v2.keras.metrics.FalseNegatives
* Class tf.compat.v2.metrics.FalseNegatives
* Class tf.keras.metrics.FalseNegatives
* Class tf.metrics.FalseNegatives

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0, 1, 1, 1] and y\_pred is [0, 1, 0, 0] then the false negatives value is 2. If the weights were specified as [0, 0, 1, 0] then the false negatives value would be 1.

If sample\_weight is given, calculates the sum of the weights of false negatives. This metric creates one local variable, accumulator that is used to keep track of the number of false negatives.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.FalseNegatives()  
m.update\_state([0, 1, 1, 1], [0, 1, 0, 0])  
print('Final result: ', m.result().numpy())  # Final result: 2

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.FalseNegatives()])

## \_\_init\_\_

\_\_init\_\_(  
    thresholds=None,  
    name=None,  
    dtype=None  
)

Creates a FalseNegatives instance.

#### Args:

* **thresholds**: (Optional) Defaults to 0.5. A float value or a python list/tuple of float threshold values in [0, 1]. A threshold is compared with prediction values to determine the truth value of predictions (i.e., above the threshold is true, below is false). One metric value is generated for each threshold value.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates the given confusion matrix condition statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.FalsePositives

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalsePositives#top_of_page)
* [Class FalsePositives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalsePositives#class_falsepositives)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalsePositives#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalsePositives#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/FalsePositives#methods)

## Class FalsePositives

Calculates the number of false positives.

### Aliases:

* Class tf.compat.v1.keras.metrics.FalsePositives
* Class tf.compat.v2.keras.metrics.FalsePositives
* Class tf.compat.v2.metrics.FalsePositives
* Class tf.keras.metrics.FalsePositives
* Class tf.metrics.FalsePositives

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0, 1, 0, 0] and y\_pred is [0, 0, 1, 1] then the false positives value is 2. If the weights were specified as [0, 0, 1, 0] then the false positives value would be 1.

If sample\_weight is given, calculates the sum of the weights of false positives. This metric creates one local variable, accumulator that is used to keep track of the number of false positives.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.FalsePositives()  
m.update\_state([0, 1, 0, 0], [0, 0, 1, 1])  
print('Final result: ', m.result().numpy())  # Final result: 2

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.FalsePositives()])

## \_\_init\_\_

\_\_init\_\_(  
    thresholds=None,  
    name=None,  
    dtype=None  
)

Creates a FalsePositives instance.

#### Args:

* **thresholds**: (Optional) Defaults to 0.5. A float value or a python list/tuple of float threshold values in [0, 1]. A threshold is compared with prediction values to determine the truth value of predictions (i.e., above the threshold is true, below is false). One metric value is generated for each threshold value.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates the given confusion matrix condition statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/get#aliases)

Aliases:

* tf.compat.v1.keras.metrics.get
* tf.compat.v2.keras.metrics.get
* tf.compat.v2.metrics.get
* tf.keras.metrics.get
* tf.metrics.get

tf.keras.metrics.get(identifier)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

# tf.keras.metrics.Hinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Hinge#top_of_page)
* [Class Hinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Hinge#class_hinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Hinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Hinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Hinge#methods)

## Class Hinge

Computes the hinge metric between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.Hinge
* Class tf.compat.v2.keras.metrics.Hinge
* Class tf.compat.v2.metrics.Hinge
* Class tf.keras.metrics.Hinge
* Class tf.metrics.Hinge

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

y\_true values are expected to be -1 or 1. If binary (0 or 1) labels are provided we will convert them to -1 or 1.

For example, if y\_true is [-1., 1., 1.], and y\_pred is [0.6, -0.7, -0.5] the hinge metric value is 1.6.

#### Usage:

m = tf.keras.metrics.Hinge()  
m.update\_state([-1., 1., 1.], [0.6, -0.7, -0.5])  
  
# result = max(0, 1-y\_true \* y\_pred) = [1.6 + 1.7 + 1.5] / 3  
  
print('Final result: ', m.result().numpy())  # Final result: 1.6

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.Hinge()])

## \_\_init\_\_

\_\_init\_\_(  
    name='hinge',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.KLDivergence

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/KLDivergence#top_of_page)
* [Class KLDivergence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/KLDivergence#class_kldivergence)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/KLDivergence#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/KLDivergence#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/KLDivergence#methods)

## Class KLDivergence

Computes Kullback-Leibler divergence metric between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.KLDivergence
* Class tf.compat.v2.keras.metrics.KLDivergence
* Class tf.compat.v2.metrics.KLDivergence
* Class tf.keras.metrics.KLDivergence
* Class tf.metrics.KLDivergence

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

metric = y\_true \* log(y\_true / y\_pred)

#### Usage:

m = tf.keras.metrics.KLDivergence()  
m.update\_state([.4, .9, .2], [.5, .8, .12])  
print('Final result: ', m.result().numpy())  # Final result: -0.043

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.KLDivergence()])

## \_\_init\_\_

\_\_init\_\_(  
    name='kullback\_leibler\_divergence',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.LogCoshError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/LogCoshError#top_of_page)
* [Class LogCoshError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/LogCoshError#class_logcosherror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/LogCoshError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/LogCoshError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/LogCoshError#methods)

## Class LogCoshError

Computes the logarithm of the hyperbolic cosine of the prediction error.

### Aliases:

* Class tf.compat.v1.keras.metrics.LogCoshError
* Class tf.compat.v2.keras.metrics.LogCoshError
* Class tf.compat.v2.metrics.LogCoshError
* Class tf.keras.metrics.LogCoshError
* Class tf.metrics.LogCoshError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

logcosh = log((exp(x) + exp(-x))/2), where x is the error (y\_pred - y\_true)

#### Usage:

m = tf.keras.metrics.LogCoshError()  
m.update\_state([0., 1., 1.], [1., 0., 1.])  
print('Final result: ', m.result().numpy())  # Final result: 0.289

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.LogCoshError()])

## \_\_init\_\_

\_\_init\_\_(  
    name='logcosh',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.Mean

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean#top_of_page)
* [Class Mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean#class_mean)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean#used_in_the_tutorials)

## Class Mean

Computes the (weighted) mean of the given values.

### Aliases:

* Class tf.compat.v1.keras.metrics.Mean
* Class tf.compat.v2.keras.metrics.Mean
* Class tf.compat.v2.metrics.Mean
* Class tf.keras.metrics.Mean
* Class tf.metrics.Mean

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

For example, if values is [1, 3, 5, 7] then the mean is 4. If the weights were specified as [1, 1, 0, 0] then the mean would be 2.

This metric creates two variables, total and count that are used to compute the average of values. This average is ultimately returned as mean which is an idempotent operation that simply divides total by count.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.Mean()  
m.update\_state([1, 3, 5, 7])  
print('Final result: ', m.result().numpy())  # Final result: 4.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.add\_metric(tf.keras.metrics.Mean(name='mean\_1')(outputs))  
model.compile('sgd', loss='mse')

## \_\_init\_\_

\_\_init\_\_(  
    name='mean',  
    dtype=None  
)

Creates a Mean instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    values,  
    sample\_weight=None  
)

Accumulates statistics for computing the reduction metric.

For example, if values is [1, 3, 5, 7] and reduction=SUM\_OVER\_BATCH\_SIZE, then the value of result() is 4. If the sample\_weight is specified as [1, 1, 0, 0] then value of result() would be 2.

#### Args:

* **values**: Per-example value.
* **sample\_weight**: Optional weighting of each example. Defaults to 1.

#### Returns:

Update op.

# tf.keras.metrics.MeanAbsoluteError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsoluteError#top_of_page)
* [Class MeanAbsoluteError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsoluteError#class_meanabsoluteerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsoluteError#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsoluteError#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsoluteError#__init__)

## Class MeanAbsoluteError

Computes the mean absolute error between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanAbsoluteError
* Class tf.compat.v2.keras.metrics.MeanAbsoluteError
* Class tf.compat.v2.metrics.MeanAbsoluteError
* Class tf.keras.metrics.MeanAbsoluteError
* Class tf.metrics.MeanAbsoluteError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

For example, if y\_true is [0., 0., 1., 1.], and y\_pred is [1., 1., 1., 0.] the mean absolute error is 3/4 (0.75).

#### Usage:

m = tf.keras.metrics.MeanAbsoluteError()  
m.update\_state([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Final result: ', m.result().numpy())  # Final result: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.MeanAbsoluteError()])

## \_\_init\_\_

\_\_init\_\_(  
    name='mean\_absolute\_error',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.MeanAbsolutePercentageError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsolutePercentageError#top_of_page)
* [Class MeanAbsolutePercentageError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsolutePercentageError#class_meanabsolutepercentageerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsolutePercentageError#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsolutePercentageError#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanAbsolutePercentageError#__init__)

## Class MeanAbsolutePercentageError

Computes the mean absolute percentage error between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanAbsolutePercentageError
* Class tf.compat.v2.keras.metrics.MeanAbsolutePercentageError
* Class tf.compat.v2.metrics.MeanAbsolutePercentageError
* Class tf.keras.metrics.MeanAbsolutePercentageError
* Class tf.metrics.MeanAbsolutePercentageError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

For example, if y\_true is [0., 0., 1., 1.], and y\_pred is [1., 1., 1., 0.] the mean absolute percentage error is 5e+08.

#### Usage:

m = tf.keras.metrics.MeanAbsolutePercentageError()  
m.update\_state([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Final result: ', m.result().numpy())  # Final result: 5e+08

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.MeanAbsolutePercentageError()])

## \_\_init\_\_

\_\_init\_\_(  
    name='mean\_absolute\_percentage\_error',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.MeanIoU

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanIoU#top_of_page)
* [Class MeanIoU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanIoU#class_meaniou)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanIoU#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanIoU#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanIoU#methods)

## Class MeanIoU

Computes the mean Intersection-Over-Union metric.

Inherits From: [Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric)

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanIoU
* Class tf.compat.v2.keras.metrics.MeanIoU
* Class tf.compat.v2.metrics.MeanIoU
* Class tf.keras.metrics.MeanIoU
* Class tf.metrics.MeanIoU

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

Mean Intersection-Over-Union is a common evaluation metric for semantic image segmentation, which first computes the IOU for each semantic class and then computes the average over classes. IOU is defined as follows: IOU = true\_positive / (true\_positive + false\_positive + false\_negative). The predictions are accumulated in a confusion matrix, weighted by sample\_weight and the metric is then calculated from it.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.MeanIoU(num\_classes=2)  
m.update\_state([0, 0, 1, 1], [0, 1, 0, 1])  
  
  # cm = [[1, 1],  
          [1, 1]]  
  # sum\_row = [2, 2], sum\_col = [2, 2], true\_positives = [1, 1]  
  # iou = true\_positives / (sum\_row + sum\_col - true\_positives))  
  # result = (1 / (2 + 2 - 1) + 1 / (2 + 2 - 1)) / 2 = 0.33  
print('Final result: ', m.result().numpy())  # Final result: 0.33

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
  'sgd',  
  loss='mse',  
  metrics=[tf.keras.metrics.MeanIoU(num\_classes=2)])

## \_\_init\_\_

\_\_init\_\_(  
    num\_classes,  
    name=None,  
    dtype=None  
)

Creates a MeanIoU instance.

#### Args:

* **num\_classes**: The possible number of labels the prediction task can have. This value must be provided, since a confusion matrix of dimension = [num\_classes, num\_classes] will be allocated.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

Compute the mean intersection-over-union via the confusion matrix.

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates the confusion matrix statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.MeanRelativeError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanRelativeError#top_of_page)
* [Class MeanRelativeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanRelativeError#class_meanrelativeerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanRelativeError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanRelativeError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanRelativeError#methods)

## Class MeanRelativeError

Computes the mean relative error by normalizing with the given values.

Inherits From: [Mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean)

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanRelativeError
* Class tf.compat.v2.keras.metrics.MeanRelativeError
* Class tf.compat.v2.metrics.MeanRelativeError
* Class tf.keras.metrics.MeanRelativeError
* Class tf.metrics.MeanRelativeError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

This metric creates two local variables, total and count that are used to compute the mean relative absolute error. This average is weighted by sample\_weight, and it is ultimately returned as mean\_relative\_error: an idempotent operation that simply divides total by count.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.MeanRelativeError(normalizer=[1, 3, 2, 3])  
m.update\_state([1, 3, 2, 3], [2, 4, 6, 8])  
  
# metric = mean(|y\_pred - y\_true| / normalizer)  
#        = mean([1, 1, 4, 5] / [1, 3, 2, 3]) = mean([1, 1/3, 2, 5/3])  
#        = 5/4 = 1.25  
print('Final result: ', m.result().numpy())  # Final result: 1.25

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
  'sgd',  
  loss='mse',  
  metrics=[tf.keras.metrics.MeanRelativeError(normalizer=[1, 3])])

## \_\_init\_\_

\_\_init\_\_(  
    normalizer,  
    name=None,  
    dtype=None  
)

Creates a MeanRelativeError instance.

#### Args:

* **normalizer**: The normalizer values with same shape as predictions.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.MeanSquaredError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredError#top_of_page)
* [Class MeanSquaredError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredError#class_meansquarederror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredError#methods)

## Class MeanSquaredError

Computes the mean squared error between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanSquaredError
* Class tf.compat.v2.keras.metrics.MeanSquaredError
* Class tf.compat.v2.metrics.MeanSquaredError
* Class tf.keras.metrics.MeanSquaredError
* Class tf.metrics.MeanSquaredError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0., 0., 1., 1.], and y\_pred is [1., 1., 1., 0.] the mean squared error is 3/4 (0.75).

#### Usage:

m = tf.keras.metrics.MeanSquaredError()  
m.update\_state([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Final result: ', m.result().numpy())  # Final result: 0.75

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.MeanSquaredError()])

## \_\_init\_\_

\_\_init\_\_(  
    name='mean\_squared\_error',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.MeanSquaredLogarithmicError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredLogarithmicError#top_of_page)
* [Class MeanSquaredLogarithmicError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredLogarithmicError#class_meansquaredlogarithmicerror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredLogarithmicError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredLogarithmicError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanSquaredLogarithmicError#methods)

## Class MeanSquaredLogarithmicError

Computes the mean squared logarithmic error between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanSquaredLogarithmicError
* Class tf.compat.v2.keras.metrics.MeanSquaredLogarithmicError
* Class tf.compat.v2.metrics.MeanSquaredLogarithmicError
* Class tf.keras.metrics.MeanSquaredLogarithmicError
* Class tf.metrics.MeanSquaredLogarithmicError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0., 0., 1., 1.], and y\_pred is [1., 1., 1., 0.] the mean squared logarithmic error is 0.36034.

#### Usage:

m = tf.keras.metrics.MeanSquaredLogarithmicError()  
m.update\_state([0., 0., 1., 1.], [1., 1., 1., 0.])  
print('Final result: ', m.result().numpy())  # Final result: 0.36034

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.MeanSquaredLogarithmicError()])

## \_\_init\_\_

\_\_init\_\_(  
    name='mean\_squared\_logarithmic\_error',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.MeanTensor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanTensor#top_of_page)
* [Class MeanTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanTensor#class_meantensor)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanTensor#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanTensor#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/MeanTensor#properties)

## Class MeanTensor

Computes the element-wise (weighted) mean of the given tensors.

Inherits From: [Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric)

### Aliases:

* Class tf.compat.v1.keras.metrics.MeanTensor
* Class tf.compat.v2.keras.metrics.MeanTensor
* Class tf.compat.v2.metrics.MeanTensor
* Class tf.keras.metrics.MeanTensor
* Class tf.metrics.MeanTensor

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

MeanTensor returns a tensor with the same shape of the input tensors. The mean value is updated by keeping local variables total and count. The total tracks the sum of the weighted values, and count stores the sum of the weighted counts.

#### Usage:

m = tf.keras.metrics.MeanTensor()  
m.update\_state([0, 1, 2, 3])  
m.update\_state([4, 5, 6, 7])  
print('Result: ', m.result().numpy())  # Result: [2, 3, 4, 5]  
m.update\_state([12, 10, 8, 6], sample\_weights= [0, 0.2, 0.5, 1])  
print('Result: ', m.result().numpy())  # Result: [2, 3.636, 4.8, 5.333]

## \_\_init\_\_

\_\_init\_\_(  
    name='mean\_tensor',  
    dtype=None  
)

Creates a MeanTensor instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Properties

### count

### total

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    values,  
    sample\_weight=None  
)

Accumulates statistics for computing the element-wise mean.

#### Args:

* **values**: Per-example value.
* **sample\_weight**: Optional weighting of each example. Defaults to 1.

#### Returns:

Update op.

# tf.keras.metrics.Metric

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric#top_of_page)
* [Class Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric#class_metric)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric#methods)

## Class Metric

Encapsulates metric logic and state.

Inherits From: [Layer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Layer)

### Aliases:

* Class tf.compat.v1.keras.metrics.Metric
* Class tf.compat.v2.keras.metrics.Metric
* Class tf.compat.v2.metrics.Metric
* Class tf.keras.metrics.Metric
* Class tf.metrics.Metric

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

#### Usage:

m = SomeMetric(...)  
for input in ...:  
  m.update\_state(input)  
print('Final result: ', m.result().numpy())

Usage with tf.keras API:

model = tf.keras.Sequential()  
model.add(tf.keras.layers.Dense(64, activation='relu'))  
model.add(tf.keras.layers.Dense(64, activation='relu'))  
model.add(tf.keras.layers.Dense(10, activation='softmax'))  
  
model.compile(optimizer=tf.compat.v1.train.RMSPropOptimizer(0.01),  
              loss=tf.keras.losses.categorical\_crossentropy,  
              metrics=[tf.keras.metrics.CategoricalAccuracy()])  
  
data = np.random.random((1000, 32))  
labels = np.random.random((1000, 10))  
  
dataset = tf.data.Dataset.from\_tensor\_slices((data, labels))  
dataset = dataset.batch(32)  
dataset = dataset.repeat()  
  
model.fit(dataset, epochs=10, steps\_per\_epoch=30)

To be implemented by subclasses: \* \_\_init\_\_(): All state variables should be created in this method by calling self.add\_weight() like: self.var = self.add\_weight(...) \* update\_state(): Has all updates to the state variables like: self.var.assign\_add(...). \* result(): Computes and returns a value for the metric from the state variables.

Example subclass implementation:

class BinaryTruePositives(tf.keras.metrics.Metric):  
  
  def \_\_init\_\_(self, name='binary\_true\_positives', \*\*kwargs):  
    super(BinaryTruePositives, self).\_\_init\_\_(name=name, \*\*kwargs)  
    self.true\_positives = self.add\_weight(name='tp', initializer='zeros')  
  
  def update\_state(self, y\_true, y\_pred, sample\_weight=None):  
    y\_true = tf.cast(y\_true, tf.bool)  
    y\_pred = tf.cast(y\_pred, tf.bool)  
  
    values = tf.logical\_and(tf.equal(y\_true, True), tf.equal(y\_pred, True))  
    values = tf.cast(values, self.dtype)  
    if sample\_weight is not None:  
      sample\_weight = tf.cast(sample\_weight, self.dtype)  
      sample\_weight = tf.broadcast\_weights(sample\_weight, values)  
      values = tf.multiply(values, sample\_weight)  
    self.true\_positives.assign\_add(tf.reduce\_sum(values))  
  
  def result(self):  
    return self.true\_positives

## \_\_init\_\_

\_\_init\_\_(  
    name=None,  
    dtype=None,  
    \*\*kwargs  
)

## Methods

### add\_weight

add\_weight(  
    name,  
    shape=(),  
    aggregation=tf.compat.v1.VariableAggregation.SUM,  
    synchronization=tf.VariableSynchronization.ON\_READ,  
    initializer=None,  
    dtype=None  
)

Adds state variable. Only for use by subclasses.

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

Computes and returns the metric value tensor.

Result computation is an idempotent operation that simply calculates the metric value using the state variables.

### update\_state

update\_state(  
    \*args,  
    \*\*kwargs  
)

Accumulates statistics for the metric.

**Note:** This function is executed as a graph function in graph mode. This means: a) Operations on the same resource are executed in textual order. This should make it easier to do things like add the updated value of a variable to another, for example. b) You don't need to worry about collecting the update ops to execute. All update ops added to the graph by this function will be executed. As a result, code should generally work the same way with graph or eager execution.

Please use tf.config.experimental\_run\_functions\_eagerly(True) to execute this function eagerly for debugging or profiling.

#### Args:

* **\*args**: \* **\*\*kwargs**: A mini-batch of inputs to the Metric.
* tf.keras.metrics.Poisson
* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Poisson#top_of_page)
* [Class Poisson](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Poisson#class_poisson)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Poisson#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Poisson#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Poisson#methods)

## Class Poisson

Computes the Poisson metric between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.Poisson
* Class tf.compat.v2.keras.metrics.Poisson
* Class tf.compat.v2.metrics.Poisson
* Class tf.keras.metrics.Poisson
* Class tf.metrics.Poisson

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

metric = y\_pred - y\_true \* log(y\_pred)

#### Usage:

m = tf.keras.metrics.Poisson()  
m.update\_state([1, 9, 2], [4, 8, 12])  
print('Final result: ', m.result().numpy())  # Final result: -4.63

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.Poisson()])

## \_\_init\_\_

\_\_init\_\_(  
    name='poisson',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.Precision

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Precision#top_of_page)
* [Class Precision](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Precision#class_precision)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Precision#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Precision#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Precision#methods)

## Class Precision

Computes the precision of the predictions with respect to the labels.

Inherits From: [Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric)

### Aliases:

* Class tf.compat.v1.keras.metrics.Precision
* Class tf.compat.v2.keras.metrics.Precision
* Class tf.compat.v2.metrics.Precision
* Class tf.keras.metrics.Precision
* Class tf.metrics.Precision

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0, 1, 1, 1] and y\_pred is [1, 0, 1, 1] then the precision value is 2/(2+1) ie. 0.66. If the weights were specified as [0, 0, 1, 0] then the precision value would be 1.

The metric creates two local variables, true\_positives and false\_positives that are used to compute the precision. This value is ultimately returned as precision, an idempotent operation that simply divides true\_positives by the sum of true\_positives and false\_positives.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

If top\_k is set, we'll calculate precision as how often on average a class among the top-k classes with the highest predicted values of a batch entry is correct and can be found in the label for that entry.

If class\_id is specified, we calculate precision by considering only the entries in the batch for which class\_id is above the threshold and/or in the top-k highest predictions, and computing the fraction of them for which class\_id is indeed a correct label.

#### Usage:

m = tf.keras.metrics.Precision()  
m.update\_state([0, 1, 1, 1], [1, 0, 1, 1])  
print('Final result: ', m.result().numpy())  # Final result: 0.66

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.Precision()])

## \_\_init\_\_

\_\_init\_\_(  
    thresholds=None,  
    top\_k=None,  
    class\_id=None,  
    name=None,  
    dtype=None  
)

Creates a Precision instance.

#### Args:

* **thresholds**: (Optional) A float value or a python list/tuple of float threshold values in [0, 1]. A threshold is compared with prediction values to determine the truth value of predictions (i.e., above the threshold is true, below is false). One metric value is generated for each threshold value. If neither thresholds nor top\_k are set, the default is to calculate precision with thresholds=0.5.
* **top\_k**: (Optional) Unset by default. An int value specifying the top-k predictions to consider when calculating precision.
* **class\_id**: (Optional) Integer class ID for which we want binary metrics. This must be in the half-open interval [0, num\_classes), where num\_classes is the last dimension of predictions.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates true positive and false positive statistics.

#### Args:

* **y\_true**: The ground truth values, with the same dimensions as y\_pred. Will be cast to bool.
* **y\_pred**: The predicted values. Each element must be in the range [0, 1].
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.Recall

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Recall#top_of_page)
* [Class Recall](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Recall#class_recall)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Recall#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Recall#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Recall#methods)

## Class Recall

Computes the recall of the predictions with respect to the labels.

Inherits From: [Metric](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Metric)

### Aliases:

* Class tf.compat.v1.keras.metrics.Recall
* Class tf.compat.v2.keras.metrics.Recall
* Class tf.compat.v2.metrics.Recall
* Class tf.keras.metrics.Recall
* Class tf.metrics.Recall

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0, 1, 1, 1] and y\_pred is [1, 0, 1, 1] then the recall value is 2/(2+1) ie. 0.66. If the weights were specified as [0, 0, 1, 0] then the recall value would be 1.

This metric creates two local variables, true\_positives and false\_negatives, that are used to compute the recall. This value is ultimately returned as recall, an idempotent operation that simply divides true\_positives by the sum of true\_positives and false\_negatives.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

If top\_k is set, recall will be computed as how often on average a class among the labels of a batch entry is in the top-k predictions.

If class\_id is specified, we calculate recall by considering only the entries in the batch for which class\_id is in the label, and computing the fraction of them for which class\_id is above the threshold and/or in the top-k predictions.

#### Usage:

m = tf.keras.metrics.Recall()  
m.update\_state([0, 1, 1, 1], [1, 0, 1, 1])  
print('Final result: ', m.result().numpy())  # Final result: 0.66

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.Recall()])

## \_\_init\_\_

\_\_init\_\_(  
    thresholds=None,  
    top\_k=None,  
    class\_id=None,  
    name=None,  
    dtype=None  
)

Creates a Recall instance.

#### Args:

* **thresholds**: (Optional) A float value or a python list/tuple of float threshold values in [0, 1]. A threshold is compared with prediction values to determine the truth value of predictions (i.e., above the threshold is true, below is false). One metric value is generated for each threshold value. If neither thresholds nor top\_k are set, the default is to calculate recall with thresholds=0.5.
* **top\_k**: (Optional) Unset by default. An int value specifying the top-k predictions to consider when calculating recall.
* **class\_id**: (Optional) Integer class ID for which we want binary metrics. This must be in the half-open interval [0, num\_classes), where num\_classes is the last dimension of predictions.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates true positive and false negative statistics.

#### Args:

* **y\_true**: The ground truth values, with the same dimensions as y\_pred. Will be cast to bool.
* **y\_pred**: The predicted values. Each element must be in the range [0, 1].
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.RootMeanSquaredError

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/RootMeanSquaredError#top_of_page)
* [Class RootMeanSquaredError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/RootMeanSquaredError#class_rootmeansquarederror)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/RootMeanSquaredError#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/RootMeanSquaredError#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/RootMeanSquaredError#methods)

## Class RootMeanSquaredError

Computes root mean squared error metric between y\_true and y\_pred.

Inherits From: [Mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Mean)

### Aliases:

* Class tf.compat.v1.keras.metrics.RootMeanSquaredError
* Class tf.compat.v2.keras.metrics.RootMeanSquaredError
* Class tf.compat.v2.metrics.RootMeanSquaredError
* Class tf.keras.metrics.RootMeanSquaredError
* Class tf.metrics.RootMeanSquaredError

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

#### Usage:

m = tf.keras.metrics.RootMeanSquaredError()  
m.update\_state([2., 4., 6.], [1., 3., 2.])  
print('Final result: ', m.result().numpy())  # Final result: 2.449

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.RootMeanSquaredError()])

## \_\_init\_\_

\_\_init\_\_(  
    name='root\_mean\_squared\_error',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates root mean squared error statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.SensitivityAtSpecificity

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity#top_of_page)
* [Class SensitivityAtSpecificity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity#class_sensitivityatspecificity)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SensitivityAtSpecificity#methods)

## Class SensitivityAtSpecificity

Computes the sensitivity at a given specificity.

### Aliases:

* Class tf.compat.v1.keras.metrics.SensitivityAtSpecificity
* Class tf.compat.v2.keras.metrics.SensitivityAtSpecificity
* Class tf.compat.v2.metrics.SensitivityAtSpecificity
* Class tf.keras.metrics.SensitivityAtSpecificity
* Class tf.metrics.SensitivityAtSpecificity

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

Sensitivity measures the proportion of actual positives that are correctly identified as such (tp / (tp + fn)). Specificity measures the proportion of actual negatives that are correctly identified as such (tn / (tn + fp)).

This metric creates four local variables, true\_positives, true\_negatives, false\_positives and false\_negatives that are used to compute the sensitivity at the given specificity. The threshold for the given specificity value is computed and used to evaluate the corresponding sensitivity.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

For additional information about specificity and sensitivity, see the following: https://en.wikipedia.org/wiki/Sensitivity\_and\_specificity

#### Usage:

m = tf.keras.metrics.SensitivityAtSpecificity(0.4, num\_thresholds=1)  
m.update\_state([0, 0, 1, 1], [0, 0.5, 0.3, 0.9])  
print('Final result: ', m.result().numpy())  # Final result: 0.5

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
    'sgd',  
    loss='mse',  
    metrics=[tf.keras.metrics.SensitivityAtSpecificity()])

## \_\_init\_\_

\_\_init\_\_(  
    specificity,  
    num\_thresholds=200,  
    name=None,  
    dtype=None  
)

Creates a SensitivityAtSpecificity instance.

#### Args:

* **specificity**: A scalar value in range [0, 1].
* **num\_thresholds**: (Optional) Defaults to 200. The number of thresholds to use for matching the given specificity.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates confusion matrix statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/serialize#aliases)

Aliases:

* tf.compat.v1.keras.metrics.serialize
* tf.compat.v2.keras.metrics.serialize
* tf.compat.v2.metrics.serialize
* tf.keras.metrics.serialize
* tf.metrics.serialize

tf.keras.metrics.serialize(metric)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

# tf.keras.metrics.SparseCategoricalAccuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy#top_of_page)
* [Class SparseCategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy#class_sparsecategoricalaccuracy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalAccuracy#used_in_the_tutorials)

## Class SparseCategoricalAccuracy

Calculates how often predictions matches integer labels.

### Aliases:

* Class tf.compat.v1.keras.metrics.SparseCategoricalAccuracy
* Class tf.compat.v2.keras.metrics.SparseCategoricalAccuracy
* Class tf.compat.v2.metrics.SparseCategoricalAccuracy
* Class tf.keras.metrics.SparseCategoricalAccuracy
* Class tf.metrics.SparseCategoricalAccuracy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

For example, if y\_true is [[2], [1]] and y\_pred is [[0.1, 0.9, 0.8], [0.05, 0.95, 0]] then the categorical accuracy is 1/2 or .5. If the weights were specified as [0.7, 0.3] then the categorical accuracy would be .3. You can provide logits of classes as y\_pred, since argmax of logits and probabilities are same.

This metric creates two local variables, total and count that are used to compute the frequency with which y\_pred matches y\_true. This frequency is ultimately returned as sparse categorical accuracy: an idempotent operation that simply divides total by count.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.SparseCategoricalAccuracy()  
m.update\_state([[2], [1]], [[0.1, 0.9, 0.8], [0.05, 0.95, 0]])  
print('Final result: ', m.result().numpy())  # Final result: 0.5

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
    'sgd',  
    loss='mse',  
    metrics=[tf.keras.metrics.SparseCategoricalAccuracy()])

## \_\_init\_\_

\_\_init\_\_(  
    name='sparse\_categorical\_accuracy',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.SparseCategoricalCrossentropy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalCrossentropy#top_of_page)
* [Class SparseCategoricalCrossentropy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalCrossentropy#class_sparsecategoricalcrossentropy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalCrossentropy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalCrossentropy#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseCategoricalCrossentropy#methods)

## Class SparseCategoricalCrossentropy

Computes the crossentropy metric between the labels and predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.SparseCategoricalCrossentropy
* Class tf.compat.v2.keras.metrics.SparseCategoricalCrossentropy
* Class tf.compat.v2.metrics.SparseCategoricalCrossentropy
* Class tf.keras.metrics.SparseCategoricalCrossentropy
* Class tf.metrics.SparseCategoricalCrossentropy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

Use this crossentropy metric when there are two or more label classes. We expect labels to be provided as integers. If you want to provide labels using one-hot representation, please use CategoricalCrossentropy metric. There should be # classes floating point values per feature for y\_pred and a single floating point value per feature for y\_true.

In the snippet below, there is a single floating point value per example for y\_true and # classesfloating pointing values per example for y\_pred. The shape of y\_true is [batch\_size] and the shape of y\_pred is [batch\_size, num\_classes].

#### Usage:

m = tf.keras.metrics.SparseCategoricalCrossentropy()  
m.update\_state(  
  [1, 2],  
  [[0.05, 0.95, 0], [0.1, 0.8, 0.1]])  
  
# y\_true = one\_hot(y\_true) = [[0, 1, 0], [0, 0, 1]]  
# logits = log(y\_pred)  
# softmax = exp(logits) / sum(exp(logits), axis=-1)  
# softmax = [[0.05, 0.95, EPSILON], [0.1, 0.8, 0.1]]  
  
# xent = -sum(y \* log(softmax), 1)  
# log(softmax) = [[-2.9957, -0.0513, -16.1181], [-2.3026, -0.2231, -2.3026]]  
# y\_true \* log(softmax) = [[0, -0.0513, 0], [0, 0, -2.3026]]  
  
# xent = [0.0513, 2.3026]  
# Reduced xent = (0.0513 + 2.3026) / 2  
  
print('Final result: ', m.result().numpy())  # Final result: 1.176

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
  'sgd',  
  loss='mse',  
  metrics=[tf.keras.metrics.SparseCategoricalCrossentropy()])

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.
* **from\_logits**: (Optional ) Whether y\_pred is expected to be a logits tensor. By default, we assume that y\_pred encodes a probability distribution.
* **axis**: (Optional) Defaults to -1. The dimension along which the metric is computed.

## \_\_init\_\_

\_\_init\_\_(  
    name='sparse\_categorical\_crossentropy',  
    dtype=None,  
    from\_logits=False,  
    axis=-1  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.SparseTopKCategoricalAccuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseTopKCategoricalAccuracy#top_of_page)
* [Class SparseTopKCategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseTopKCategoricalAccuracy#class_sparsetopkcategoricalaccuracy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseTopKCategoricalAccuracy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseTopKCategoricalAccuracy#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SparseTopKCategoricalAccuracy#methods)

## Class SparseTopKCategoricalAccuracy

Computes how often integer targets are in the top K predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.SparseTopKCategoricalAccuracy
* Class tf.compat.v2.keras.metrics.SparseTopKCategoricalAccuracy
* Class tf.compat.v2.metrics.SparseTopKCategoricalAccuracy
* Class tf.keras.metrics.SparseTopKCategoricalAccuracy
* Class tf.metrics.SparseTopKCategoricalAccuracy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

#### Usage:

m = tf.keras.metrics.SparseTopKCategoricalAccuracy()  
m.update\_state([2, 1], [[0.1, 0.9, 0.8], [0.05, 0.95, 0]])  
print('Final result: ', m.result().numpy())  # Final result: 1.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
  'sgd',  
  metrics=[tf.keras.metrics.SparseTopKCategoricalAccuracy()])

## \_\_init\_\_

\_\_init\_\_(  
    k=5,  
    name='sparse\_top\_k\_categorical\_accuracy',  
    dtype=None  
)

Creates a SparseTopKCategoricalAccuracy instance.

#### Args:

* **k**: (Optional) Number of top elements to look at for computing accuracy. Defaults to 5.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.sparse\_categorical\_accuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/sparse_categorical_accuracy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/sparse_categorical_accuracy#aliases)

Aliases:

* tf.compat.v1.keras.metrics.sparse\_categorical\_accuracy
* tf.compat.v2.keras.metrics.sparse\_categorical\_accuracy
* tf.compat.v2.metrics.sparse\_categorical\_accuracy
* tf.keras.metrics.sparse\_categorical\_accuracy
* tf.metrics.sparse\_categorical\_accuracy

tf.keras.metrics.sparse\_categorical\_accuracy(  
    y\_true,  
    y\_pred  
)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

tf.keras.metrics.sparse\_top\_k\_categorical\_accuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/sparse_top_k_categorical_accuracy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/sparse_top_k_categorical_accuracy#aliases)

Aliases:

* tf.compat.v1.keras.metrics.sparse\_top\_k\_categorical\_accuracy
* tf.compat.v2.keras.metrics.sparse\_top\_k\_categorical\_accuracy
* tf.compat.v2.metrics.sparse\_top\_k\_categorical\_accuracy
* tf.keras.metrics.sparse\_top\_k\_categorical\_accuracy
* tf.metrics.sparse\_top\_k\_categorical\_accuracy

tf.keras.metrics.sparse\_top\_k\_categorical\_accuracy(  
    y\_true,  
    y\_pred,  
    k=5  
)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

# tf.keras.metrics.SpecificityAtSensitivity

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SpecificityAtSensitivity#top_of_page)
* [Class SpecificityAtSensitivity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SpecificityAtSensitivity#class_specificityatsensitivity)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SpecificityAtSensitivity#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SpecificityAtSensitivity#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SpecificityAtSensitivity#methods)

## Class SpecificityAtSensitivity

Computes the specificity at a given sensitivity.

### Aliases:

* Class tf.compat.v1.keras.metrics.SpecificityAtSensitivity
* Class tf.compat.v2.keras.metrics.SpecificityAtSensitivity
* Class tf.compat.v2.metrics.SpecificityAtSensitivity
* Class tf.keras.metrics.SpecificityAtSensitivity
* Class tf.metrics.SpecificityAtSensitivity

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

Sensitivity measures the proportion of actual positives that are correctly identified as such (tp / (tp + fn)). Specificity measures the proportion of actual negatives that are correctly identified as such (tn / (tn + fp)).

This metric creates four local variables, true\_positives, true\_negatives, false\_positives and false\_negatives that are used to compute the specificity at the given sensitivity. The threshold for the given sensitivity value is computed and used to evaluate the corresponding specificity.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

For additional information about specificity and sensitivity, see the following: https://en.wikipedia.org/wiki/Sensitivity\_and\_specificity

#### Usage:

m = tf.keras.metrics.SpecificityAtSensitivity(0.8, num\_thresholds=1)  
m.update\_state([0, 0, 1, 1], [0, 0.5, 0.3, 0.9])  
print('Final result: ', m.result().numpy())  # Final result: 1.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile(  
    'sgd',  
    loss='mse',  
    metrics=[tf.keras.metrics.SpecificityAtSensitivity()])

## \_\_init\_\_

\_\_init\_\_(  
    sensitivity,  
    num\_thresholds=200,  
    name=None,  
    dtype=None  
)

Creates a SpecificityAtSensitivity instance.

#### Args:

* **sensitivity**: A scalar value in range [0, 1].
* **num\_thresholds**: (Optional) Defaults to 200. The number of thresholds to use for matching the given specificity.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates confusion matrix statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.SquaredHinge

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SquaredHinge#top_of_page)
* [Class SquaredHinge](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SquaredHinge#class_squaredhinge)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SquaredHinge#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SquaredHinge#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/SquaredHinge#methods)

## Class SquaredHinge

Computes the squared hinge metric between y\_true and y\_pred.

### Aliases:

* Class tf.compat.v1.keras.metrics.SquaredHinge
* Class tf.compat.v2.keras.metrics.SquaredHinge
* Class tf.compat.v2.metrics.SquaredHinge
* Class tf.keras.metrics.SquaredHinge
* Class tf.metrics.SquaredHinge

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

y\_true values are expected to be -1 or 1. If binary (0 or 1) labels are provided we will convert them to -1 or 1.

For example, if y\_true is [-1., 1., 1.], and y\_pred is [0.6, -0.7, -0.5] the squared hinge metric value is 2.6.

#### Usage:

m = tf.keras.metrics.SquaredHinge()  
m.update\_state([-1., 1., 1.], [0.6, -0.7, -0.5])  
  
# result = max(0, 1-y\_true \* y\_pred) = [1.6^2 + 1.7^2 + 1.5^2] / 3  
  
print('Final result: ', m.result().numpy())  # Final result: 2.6

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.SquaredHinge()])

## \_\_init\_\_

\_\_init\_\_(  
    name='squared\_hinge',  
    dtype=None  
)

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.Sum

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Sum#top_of_page)
* [Class Sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Sum#class_sum)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Sum#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Sum#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/Sum#methods)

## Class Sum

Computes the (weighted) sum of the given values.

### Aliases:

* Class tf.compat.v1.keras.metrics.Sum
* Class tf.compat.v2.keras.metrics.Sum
* Class tf.compat.v2.metrics.Sum
* Class tf.keras.metrics.Sum
* Class tf.metrics.Sum

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if values is [1, 3, 5, 7] then the sum is 16. If the weights were specified as [1, 1, 0, 0] then the sum would be 4.

This metric creates one variable, total, that is used to compute the sum of values. This is ultimately returned as sum.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.Sum()  
m.update\_state([1, 3, 5, 7])  
print('Final result: ', m.result().numpy())  # Final result: 16.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.add\_metric(tf.keras.metrics.Sum(name='sum\_1')(outputs))  
model.compile('sgd', loss='mse')

## \_\_init\_\_

\_\_init\_\_(  
    name='sum',  
    dtype=None  
)

Creates a Sum instance.

#### Args:

* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    values,  
    sample\_weight=None  
)

Accumulates statistics for computing the reduction metric.

For example, if values is [1, 3, 5, 7] and reduction=SUM\_OVER\_BATCH\_SIZE, then the value of result() is 4. If the sample\_weight is specified as [1, 1, 0, 0] then value of result() would be 2.

#### Args:

* **values**: Per-example value.
* **sample\_weight**: Optional weighting of each example. Defaults to 1.

#### Returns:

Update op.

# tf.keras.metrics.TopKCategoricalAccuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TopKCategoricalAccuracy#top_of_page)
* [Class TopKCategoricalAccuracy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TopKCategoricalAccuracy#class_topkcategoricalaccuracy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TopKCategoricalAccuracy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TopKCategoricalAccuracy#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TopKCategoricalAccuracy#methods)

## Class TopKCategoricalAccuracy

Computes how often targets are in the top K predictions.

### Aliases:

* Class tf.compat.v1.keras.metrics.TopKCategoricalAccuracy
* Class tf.compat.v2.keras.metrics.TopKCategoricalAccuracy
* Class tf.compat.v2.metrics.TopKCategoricalAccuracy
* Class tf.keras.metrics.TopKCategoricalAccuracy
* Class tf.metrics.TopKCategoricalAccuracy

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

#### Usage:

m = tf.keras.metrics.TopKCategoricalAccuracy()  
m.update\_state([[0, 0, 1], [0, 1, 0]], [[0.1, 0.9, 0.8], [0.05, 0.95, 0]])  
print('Final result: ', m.result().numpy())  # Final result: 1.0

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', metrics=[tf.keras.metrics.TopKCategoricalAccuracy()])

## \_\_init\_\_

\_\_init\_\_(  
    k=5,  
    name='top\_k\_categorical\_accuracy',  
    dtype=None  
)

Creates a TopKCategoricalAccuracy instance.

#### Args:

* **k**: (Optional) Number of top elements to look at for computing accuracy. Defaults to 5.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

Resets all of the metric state variables.

This function is called between epochs/steps, when a metric is evaluated during training.

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates metric statistics.

y\_true and y\_pred should have the same shape.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

tf.keras.metrics.top\_k\_categorical\_accuracy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/top_k_categorical_accuracy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/top_k_categorical_accuracy#aliases)

Aliases:

* tf.compat.v1.keras.metrics.top\_k\_categorical\_accuracy
* tf.compat.v2.keras.metrics.top\_k\_categorical\_accuracy
* tf.compat.v2.metrics.top\_k\_categorical\_accuracy
* tf.keras.metrics.top\_k\_categorical\_accuracy
* tf.metrics.top\_k\_categorical\_accuracy

tf.keras.metrics.top\_k\_categorical\_accuracy(  
    y\_true,  
    y\_pred,  
    k=5  
)

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

# tf.keras.metrics.TrueNegatives

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TrueNegatives#top_of_page)
* [Class TrueNegatives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TrueNegatives#class_truenegatives)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TrueNegatives#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TrueNegatives#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TrueNegatives#methods)

## Class TrueNegatives

Calculates the number of true negatives.

### Aliases:

* Class tf.compat.v1.keras.metrics.TrueNegatives
* Class tf.compat.v2.keras.metrics.TrueNegatives
* Class tf.compat.v2.metrics.TrueNegatives
* Class tf.keras.metrics.TrueNegatives
* Class tf.metrics.TrueNegatives

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0, 1, 0, 0] and y\_pred is [1, 1, 0, 0] then the true negatives value is 2. If the weights were specified as [0, 0, 1, 0] then the true negatives value would be 1.

If sample\_weight is given, calculates the sum of the weights of true negatives. This metric creates one local variable, accumulator that is used to keep track of the number of true negatives.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.TrueNegatives()  
m.update\_state([0, 1, 0, 0], [1, 1, 0, 0])  
print('Final result: ', m.result().numpy())  # Final result: 2

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.TrueNegatives()])

## \_\_init\_\_

\_\_init\_\_(  
    thresholds=None,  
    name=None,  
    dtype=None  
)

Creates a TrueNegatives instance.

#### Args:

* **thresholds**: (Optional) Defaults to 0.5. A float value or a python list/tuple of float threshold values in [0, 1]. A threshold is compared with prediction values to determine the truth value of predictions (i.e., above the threshold is true, below is false). One metric value is generated for each threshold value.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates the given confusion matrix condition statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

# tf.keras.metrics.TruePositives

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TruePositives#top_of_page)
* [Class TruePositives](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TruePositives#class_truepositives)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TruePositives#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TruePositives#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/metrics/TruePositives#methods)

## Class TruePositives

Calculates the number of true positives.

### Aliases:

* Class tf.compat.v1.keras.metrics.TruePositives
* Class tf.compat.v2.keras.metrics.TruePositives
* Class tf.compat.v2.metrics.TruePositives
* Class tf.keras.metrics.TruePositives
* Class tf.metrics.TruePositives

Defined in [python/keras/metrics.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/metrics.py).

For example, if y\_true is [0, 1, 1, 1] and y\_pred is [1, 0, 1, 1] then the true positives value is 2. If the weights were specified as [0, 0, 1, 0] then the true positives value would be 1.

If sample\_weight is given, calculates the sum of the weights of true positives. This metric creates one local variable, true\_positives that is used to keep track of the number of true positives.

If sample\_weight is None, weights default to 1. Use sample\_weight of 0 to mask values.

#### Usage:

m = tf.keras.metrics.TruePositives()  
m.update\_state([0, 1, 1, 1], [1, 0, 1, 1])  
print('Final result: ', m.result().numpy())  # Final result: 2

Usage with tf.keras API:

model = tf.keras.Model(inputs, outputs)  
model.compile('sgd', loss='mse', metrics=[tf.keras.metrics.TruePositives()])

## \_\_init\_\_

\_\_init\_\_(  
    thresholds=None,  
    name=None,  
    dtype=None  
)

Creates a TruePositives instance.

#### Args:

* **thresholds**: (Optional) Defaults to 0.5. A float value or a python list/tuple of float threshold values in [0, 1]. A threshold is compared with prediction values to determine the truth value of predictions (i.e., above the threshold is true, below is false). One metric value is generated for each threshold value.
* **name**: (Optional) string name of the metric instance.
* **dtype**: (Optional) data type of the metric result.

## Methods

### reset\_states

reset\_states()

### result

result()

### update\_state

update\_state(  
    y\_true,  
    y\_pred,  
    sample\_weight=None  
)

Accumulates the given confusion matrix condition statistics.

#### Args:

* **y\_true**: The ground truth values.
* **y\_pred**: The predicted values.
* **sample\_weight**: Optional weighting of each example. Defaults to 1. Can be a Tensor whose rank is either 0, or the same rank as y\_true, and must be broadcastable to y\_true.

#### Returns:

Update op.

Module: tf.keras.mixed\_precision

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision#aliases)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision#modules)

Public API for tf.keras.mixed\_precision namespace.

Aliases:

* Module tf.compat.v2.keras.mixed\_precision
* Module tf.keras.mixed\_precision

Defined in [python/keras/api/\_v2/keras/mixed\_precision/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/mixed_precision/__init__.py).

Modules

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental) module: Mixed precision API.

# tf.keras.mixed\_precision.experimental.global\_policy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/global_policy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/global_policy#aliases)

Returns the global Policy.

### Aliases:

* tf.compat.v1.keras.mixed\_precision.experimental.global\_policy
* tf.compat.v2.keras.mixed\_precision.experimental.global\_policy
* tf.keras.mixed\_precision.experimental.global\_policy

tf.keras.mixed\_precision.experimental.global\_policy()

Defined in [python/keras/mixed\_precision/experimental/policy.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/mixed_precision/experimental/policy.py).

The global policy is the default policy used for layers, if no policy is passed to the layer constructor. When TensorFlow starts, the global policy is set to an "infer" policy, and can be changed with set\_policy.

#### Returns:

The global Policy.

# tf.keras.mixed\_precision.experimental.LossScaleOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/LossScaleOptimizer#top_of_page)
* [Class LossScaleOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/LossScaleOptimizer#class_lossscaleoptimizer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/LossScaleOptimizer#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/LossScaleOptimizer#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/LossScaleOptimizer#properties)

## Class LossScaleOptimizer

An optimizer that applies loss scaling.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.mixed\_precision.experimental.LossScaleOptimizer
* Class tf.compat.v2.keras.mixed\_precision.experimental.LossScaleOptimizer
* Class tf.keras.mixed\_precision.experimental.LossScaleOptimizer

Defined in [python/keras/mixed\_precision/experimental/loss\_scale\_optimizer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/mixed_precision/experimental/loss_scale_optimizer.py).

Loss scaling is a process that multiplies the loss by a multiplier called the loss scale, and divides each gradient by the same multiplier. The pseudocode for this process is:

loss = ...  
loss \*= loss\_scale  
grads = gradients(loss, vars)  
grads /= loss\_scale

Mathematically, loss scaling has no effect, but can help avoid numerical underflow in intermediate gradients when float16 tensors are used. By multiplying the loss, each intermediate gradient will have the same multiplier applied.

The loss scale can either be a fixed constant, chosen by the user, or be dynamically determined. Dynamically determining the loss scale is convenient as a loss scale does not have to be explicitly chosen. However it reduces performance.

This optimizer wraps another optimizer and applies loss scaling to it via a LossScale. Loss scaling is applied whenever gradients are computed, either through minimize() or get\_gradients().

## \_\_init\_\_

\_\_init\_\_(  
    opt,  
    loss\_scale  
)

Initializes this loss scale optimizer.

#### Args:

* **opt**: The Optimizer instance to wrap.
* **loss\_scale**: The loss scale to scale the loss and gradients. This can either be an int/float to use a fixed loss scale, the string "dynamic" to use dynamic loss scaling, or an instance of a LossScale. The string "dynamic" equivalent to passing DynamicLossScale(), and passing an int/float is equivalent to passing a FixedLossScale with the given loss scale.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### learning\_rate

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

### from\_config

@classmethod  
from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.mixed\_precision.experimental.Policy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/Policy#top_of_page)
* [Class Policy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/Policy#class_policy)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/Policy#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/Policy#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/Policy#properties)

## Class Policy

A mixed precision policy for a Keras layer.

### Aliases:

* Class tf.compat.v1.keras.mixed\_precision.experimental.Policy
* Class tf.compat.v2.keras.mixed\_precision.experimental.Policy
* Class tf.keras.mixed\_precision.experimental.Policy

Defined in [python/keras/mixed\_precision/experimental/policy.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/mixed_precision/experimental/policy.py).

A mixed precision policy determines the floating-point dtype that Keras layers should create variables in. For non-default policies, if the variable dtype does not match the input dtype, variables will automatically be casted to the input dtype to avoid type errors. Policies can be passed to the 'dtype' argument of layer constructors, or a global policy can be set with 'set\_policy'.

In the near future, policies will also determine the computation dtype of layers, as well as the loss scaling algorithm.

Policies are intended to enable mixed precision training, which require using float32 variables and [b]float16 computations for most layers. The term "mixed precision" refers to the use of both float16 (or bfloat16) and float32 in a model. See https://arxiv.org/abs/1710.03740 for more information on mixed precision training.

Policies are constructed by passing a string to the name constructor argument. name determines the behavior of the policy. Currently, name can be one of the following values.

* 'infer': Infer the variable and computation dtypes from the input dtype. This is the default behavior.
* 'infer\_float32\_vars': Infer the computation dtypes from the input dtype, but create variables in float32. Variables will be casted to the computation dtype. This is intended to enable mixed precision. Users can cast tensors to float16 before passing them to a layer, which causes the layer to run it's computation in float16 while keeping variables in float32.

To use mixed precision in a model, the 'infer\_float32\_vars' policy can be used alongside float16 input tensors, which results in float16 computations and float32 variables. For example:

tf.keras.mixed\_precision.experimental.set\_policy('infer\_float32\_vars')  
model = tf.keras.models.Sequential(  
    tf.keras.layers.Input((100,), dtype='float16'),  
    tf.keras.layers.Dense(10),  
    tf.keras.layers.Dense(10),  
    tf.keras.layers.Lambda(lambda x: tf.cast(x, 'float32')),  
    tf.keras.layers.Activation('Softmax')  
)

Alternatively, the policy can be passed to individual layers instead of setting the global policy with set\_policy:

policy = tf.keras.mixed\_precision.experimental.Policy('infer\_float32\_vars')  
model = tf.keras.models.Sequential(  
    tf.keras.layers.Input((100,), dtype='float16'),  
    tf.keras.layers.Dense(10, dtype=policy),  
    tf.keras.layers.Dense(10, dtype=policy),  
    tf.keras.layers.Lambda(lambda x: tf.cast(x, 'float32')),  
    tf.keras.layers.Activation('Softmax')  
)

Note that a LossScaleOptimizer should also be used for mixed precision models to avoid numerical underflow. See LossScaleOptimizer.

## \_\_init\_\_

\_\_init\_\_(name)

## Properties

### default\_variable\_dtype

Returns the default variable dtype of this policy.

This is the dtype layers will create their variables in, unless a layer explicit chooses a different dtype. Layers will cast variables to the appropriate dtype to avoid type errors.

#### Returns:

The default variable dtype of this policy, or None if the default variable dtype should be derived from the inputs.

### name

Returns the name of the policy: "infer" or "infer\_float32\_vars.

### should\_cast\_variables

Returns true if variables should be casted.

tf.keras.mixed\_precision.experimental.set\_policy

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/set_policy#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/mixed_precision/experimental/set_policy#aliases)

Sets the global Policy.

Aliases:

* tf.compat.v1.keras.mixed\_precision.experimental.set\_policy
* tf.compat.v2.keras.mixed\_precision.experimental.set\_policy
* tf.keras.mixed\_precision.experimental.set\_policy

tf.keras.mixed\_precision.experimental.set\_policy(policy)

Defined in [python/keras/mixed\_precision/experimental/policy.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/mixed_precision/experimental/policy.py).

Module: tf.keras.models

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models#functions)

Code for model cloning, plus model-related API entries.

Aliases:

* Module tf.compat.v2.keras.models
* Module tf.keras.models

Defined in [python/keras/api/\_v2/keras/models/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/models/__init__.py).

Classes

[class Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model): Model groups layers into an object with training and inference features.

[class Sequential](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Sequential): Linear stack of layers.

Functions

[clone\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/clone_model): Clone any Model instance.

[load\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/load_model): Loads a model saved via save\_model.

[model\_from\_config(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_config): Instantiates a Keras model from its config.

[model\_from\_json(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_json): Parses a JSON model configuration file and returns a model instance.

[model\_from\_yaml(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_yaml): Parses a yaml model configuration file and returns a model instance.

[save\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/save_model): Saves a model as a TensorFlow SavedModel or HDF5 file.

# tf.keras.models.clone\_model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/clone_model#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/clone_model#aliases)

Clone any Model instance.

### Aliases:

* tf.compat.v1.keras.models.clone\_model
* tf.compat.v2.keras.models.clone\_model
* tf.keras.models.clone\_model

tf.keras.models.clone\_model(  
    model,  
    input\_tensors=None,  
    clone\_function=None  
)

Defined in [python/keras/models.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/models.py).

Model cloning is similar to calling a model on new inputs, except that it creates new layers (and thus new weights) instead of sharing the weights of the existing layers.

#### Arguments:

* **model**: Instance of Model (could be a functional model or a Sequential model).
* **input\_tensors**: optional list of input tensors or InputLayer objects to build the model upon. If not provided, placeholders will be created.
* **clone\_function**: Callable to be used to clone each layer in the target model (except InputLayer instances). It takes as argument the layer instance to be cloned, and returns the corresponding layer instance to be used in the model copy. If unspecified, this callable defaults to the following serialization/deserialization function: lambda layer: layer.\_\_class\_\_.from\_config(layer.get\_config()). By passing a custom callable, you can customize your copy of the model, e.g. by wrapping certain layers of interest (you might want to replace all LSTM instances with equivalent Bidirectional(LSTM(...)) instances, for example).

#### Returns:

An instance of Model reproducing the behavior of the original model, on top of new inputs tensors, using newly instantiated weights. The cloned model might behave differently from the original model if a custom clone\_function modifies the layer.

#### Raises:

* **ValueError**: in case of invalid model argument value.

# tf.keras.models.load\_model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/load_model#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/load_model#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/load_model#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/load_model#used_in_the_tutorials)

Loads a model saved via save\_model.

### Aliases:

* tf.compat.v1.keras.models.load\_model
* tf.compat.v2.keras.models.load\_model
* tf.keras.models.load\_model

tf.keras.models.load\_model(  
    filepath,  
    custom\_objects=None,  
    compile=True  
)

Defined in [python/keras/saving/save.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/save.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

### Used in the tutorials:

* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)

#### Arguments:

* **filepath**: One of the following:
  + String, path to the saved model
  + h5py.File object from which to load the model
* **custom\_objects**: Optional dictionary mapping names (strings) to custom classes or functions to be considered during deserialization.
* **compile**: Boolean, whether to compile the model after loading.

#### Returns:

A Keras model instance. If an optimizer was found as part of the saved model, the model is already compiled. Otherwise, the model is uncompiled and a warning will be displayed. When compile is set to False, the compilation is omitted without any warning.

#### Raises:

* **ImportError**: if loading from an hdf5 file and h5py is not available.
* **IOError**: In case of an invalid savefile.

# tf.keras.models.model\_from\_config

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_config#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_config#aliases)

Instantiates a Keras model from its config.

### Aliases:

* tf.compat.v1.keras.models.model\_from\_config
* tf.compat.v2.keras.models.model\_from\_config
* tf.keras.models.model\_from\_config

tf.keras.models.model\_from\_config(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/saving/model\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/model_config.py).

#### Arguments:

* **config**: Configuration dictionary.
* **custom\_objects**: Optional dictionary mapping names (strings) to custom classes or functions to be considered during deserialization.

#### Returns:

A Keras model instance (uncompiled).

#### Raises:

* **TypeError**: if config is not a dictionary.

# tf.keras.models.model\_from\_json

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_json#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_json#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_json#used_in_the_guide)

Parses a JSON model configuration file and returns a model instance.

### Aliases:

* tf.compat.v1.keras.models.model\_from\_json
* tf.compat.v2.keras.models.model\_from\_json
* tf.keras.models.model\_from\_json

tf.keras.models.model\_from\_json(  
    json\_string,  
    custom\_objects=None  
)

Defined in [python/keras/saving/model\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/model_config.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)

#### Arguments:

* **json\_string**: JSON string encoding a model configuration.
* **custom\_objects**: Optional dictionary mapping names (strings) to custom classes or functions to be considered during deserialization.

#### Returns:

A Keras model instance (uncompiled).

# tf.keras.models.model\_from\_yaml

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_yaml#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_yaml#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/model_from_yaml#used_in_the_guide)

Parses a yaml model configuration file and returns a model instance.

### Aliases:

* tf.compat.v1.keras.models.model\_from\_yaml
* tf.compat.v2.keras.models.model\_from\_yaml
* tf.keras.models.model\_from\_yaml

tf.keras.models.model\_from\_yaml(  
    yaml\_string,  
    custom\_objects=None  
)

Defined in [python/keras/saving/model\_config.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/model_config.py).

### Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)

#### Arguments:

* **yaml\_string**: YAML string encoding a model configuration.
* **custom\_objects**: Optional dictionary mapping names (strings) to custom classes or functions to be considered during deserialization.

#### Returns:

A Keras model instance (uncompiled).

#### Raises:

* **ImportError**: if yaml module is not found.

# tf.keras.models.save\_model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/save_model#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/save_model#aliases)

Saves a model as a TensorFlow SavedModel or HDF5 file.

### Aliases:

* tf.compat.v1.keras.models.save\_model
* tf.compat.v2.keras.models.save\_model
* tf.keras.models.save\_model

tf.keras.models.save\_model(  
    model,  
    filepath,  
    overwrite=True,  
    include\_optimizer=True,  
    save\_format=None  
)

Defined in [python/keras/saving/save.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/saving/save.py).

The saved model contains: - the model's configuration (topology) - the model's weights - the model's optimizer's state (if any)

Thus the saved model can be reinstantiated in the exact same state, without any of the code used for model definition or training.

SavedModel serialization (not yet added)

The SavedModel serialization path uses [tf.saved\_model.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save) to save the model and all trackable objects attached to the model (e.g. layers and variables). @tf.function-decorated methods are also saved. Additional trackable objects and functions are added to the SavedModel to allow the model to be loaded back as a Keras Model object.

#### Arguments:

* **model**: Keras model instance to be saved.
* **filepath**: One of the following:
  + String, path where to save the model
  + h5py.File object where to save the model
* **overwrite**: Whether we should overwrite any existing model at the target location, or instead ask the user with a manual prompt.
* **include\_optimizer**: If True, save optimizer's state together.
* **save\_format**: Either 'tf' or 'h5', indicating whether to save the model to Tensorflow SavedModel or HDF5. Defaults to 'tf' in TF 2.X, and 'h5' in TF 1.X.

#### Raises:

* **ImportError**: If save format is hdf5, and h5py is not available.

Module: tf.keras.optimizers

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers#aliases)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers#modules)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers#functions)

Built-in optimizer classes.

Aliases:

* Module tf.compat.v2.keras.optimizers
* Module tf.compat.v2.optimizers
* Module tf.keras.optimizers
* Module tf.optimizers

Defined in [python/keras/api/\_v2/keras/optimizers/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/optimizers/__init__.py).

Modules

[schedules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules) module: Public API for tf.keras.optimizers.schedules namespace.

Classes

[class Adadelta](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adadelta): Optimizer that implements the Adadelta algorithm.

[class Adagrad](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adagrad): Optimizer that implements the Adagrad algorithm.

[class Adam](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam): Optimizer that implements the Adam algorithm.

[class Adamax](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adamax): Optimizer that implements the Adamax algorithm.

[class Ftrl](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Ftrl): Optimizer that implements the FTRL algorithm.

[class Nadam](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Nadam): Optimizer that implements the NAdam algorithm.

[class Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer): Updated base class for optimizers.

[class RMSprop](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/RMSprop): Optimizer that implements the RMSprop algorithm.

[class SGD](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD): Stochastic gradient descent and momentum optimizer.

Functions

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/deserialize): Inverse of the serialize function.

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/get): Retrieves a Keras Optimizer instance.

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/serialize)

# tf.keras.optimizers.Adadelta

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adadelta#top_of_page)
* [Class Adadelta](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adadelta#class_adadelta)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adadelta#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adadelta#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adadelta#properties)

## Class Adadelta

Optimizer that implements the Adadelta algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.Adadelta
* Class tf.compat.v2.keras.optimizers.Adadelta
* Class tf.compat.v2.optimizers.Adadelta
* Class tf.keras.optimizers.Adadelta
* Class tf.optimizers.Adadelta

Defined in [python/keras/optimizer\_v2/adadelta.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/adadelta.py).

Adadelta optimization is a stochastic gradient descent method that is based on adaptive learning rate per dimension to address two drawbacks: 1) the continual decay of learning rates throughout training 2) the need for a manually selected global learning rate

Two accumulation steps are required: 1) the accumulation of gradients squared, 2) the accumulation of updates squared.

#### Initialization:

E[g2]0:=0(Initialize gradient 2nd order moment vector)

E[Δx2]0:=0(Initialize 2nd order variable update)

t:=t+1

E[g2]t:=ρ∗E[g2]t−1+(1−ρ)∗g2

Δxt=−RMS[Δx]t−1∗gt/RMS[g]t

E[Δx2]t:=ρ∗E[Δx2]t−1+(1−ρ)∗Δxt2

$$xt := x{t-1} + \Delta x\_{t}

References See [M. D. Zeiler](http://arxiv.org/abs/1212.5701) ([pdf](http://arxiv.org/pdf/1212.5701v1.pdf))

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    rho=0.95,  
    epsilon=1e-07,  
    name='Adadelta',  
    \*\*kwargs  
)

Construct a new Adadelta optimizer.

Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. This way, Adadelta continues learning even when many updates have been done. Compared to Adagrad, in the original version of Adadelta you don't have to set an initial learning rate. In this version, initial learning rate can be set, as in most other Keras optimizers.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate. To match the exact form in the original paper use 1.0.
* **rho**: A Tensor or a floating point value. The decay rate.
* **epsilon**: A Tensor or a floating point value. A constant epsilon used to better conditioning the grad update.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Adadelta".
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

#### Eager Compatibility

When eager execution is enabled, learning\_rate, rho, and epsilon can each be a callable that takes no **Arguments** and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.Adagrad

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adagrad#top_of_page)
* [Class Adagrad](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adagrad#class_adagrad)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adagrad#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adagrad#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adagrad#properties)

## Class Adagrad

Optimizer that implements the Adagrad algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.Adagrad
* Class tf.compat.v2.keras.optimizers.Adagrad
* Class tf.compat.v2.optimizers.Adagrad
* Class tf.keras.optimizers.Adagrad
* Class tf.optimizers.Adagrad

Defined in [python/keras/optimizer\_v2/adagrad.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/adagrad.py).

Adagrad is an optimizer with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the updates.

#### Initialization:

accumg0:=initial\_accumulator\_value

#### Update step:

t:=t+1

accumgt:=accumgt−1+g2

θt:=θt−1−lr∗g/(accumgt+ϵ)

#### References:

* [Paper](http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf).
* [Introduction](https://ppasupat.github.io/a9online/uploads/proximal_notes.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    initial\_accumulator\_value=0.1,  
    epsilon=1e-07,  
    name='Adagrad',  
    \*\*kwargs  
)

Construct a new Adagrad optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **initial\_accumulator\_value**: A floating point value. Starting value for the accumulators, must be positive.
* **epsilon**: A floating point value. Starting value for the accumulators, must be positive.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Adagrad".
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

#### Raises:

* **ValueError**: If the initial\_accumulator\_value or epsilon is invalid.

#### Eager Compatibility

When eager execution is enabled, learning\_rate can be a callable that takes no **Arguments** and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

@classmethod  
from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.Adam

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam#top_of_page)
* [Class Adam](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam#class_adam)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam#used_in_the_tutorials)

## Class Adam

Optimizer that implements the Adam algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.Adam
* Class tf.compat.v2.keras.optimizers.Adam
* Class tf.compat.v2.optimizers.Adam
* Class tf.keras.optimizers.Adam
* Class tf.optimizers.Adam

Defined in [python/keras/optimizer\_v2/adam.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/adam.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Get started with TensorFlow 2.0 for experts](https://www.tensorflow.org/beta/tutorials/quickstart/advanced)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. According to the paper [Adam: A Method for Stochastic Optimization. Kingma et al., 2014](http://arxiv.org/abs/1412.6980), the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

For AMSGrad see [On The Convergence Of Adam And Beyond. Reddi et al., 5-8](https://openreview.net/pdf?id=ryQu7f-RZ).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    beta\_1=0.9,  
    beta\_2=0.999,  
    epsilon=1e-07,  
    amsgrad=False,  
    name='Adam',  
    \*\*kwargs  
)

Construct a new Adam optimizer.

If amsgrad = False: Initialization:

m0:=0(Initialize initial 1st moment vector)

v0:=0(Initialize initial 2nd moment vector)

t:=0(Initialize timestep)

The update rule for variable with gradient g uses an optimization described at the end of section 2 of the paper:

t:=t+1

lrt:=learning\\_rate∗1−beta2t/(1−beta1t)

mt:=beta1∗mt−1+(1−beta1)∗g

vt:=beta2∗vt−1+(1−beta2)∗g∗g

variable:=variable−lrt∗mt/(vt+ϵ)

If amsgrad = True: Initialization:

m0:=0(Initialize initial 1st moment vector)

v0:=0(Initialize initial 2nd moment vector)

vhat0:=0(Initialize initial 2nd moment vector)

t:=0(Initialize timestep)

The update rule for variable with gradient g uses an optimization described at the end of section 2 of the paper:

t:=t+1

lrt:=learning\\_rate∗1−beta2t/(1−beta1t)

mt:=beta1∗mt−1+(1−beta1)∗g

vt:=beta2∗vt−1+(1−beta2)∗g∗g

vhatt:=max(vhatt−1,vt)

variable := variable - lr\_t \* m\_t / (\sqrt{v\_hat\_t} + \epsilon)$$

The default value of 1e-7 for epsilon might not be a good default in general. For example, when training an Inception network on ImageNet a current good choice is 1.0 or 0.1. Note that since AdamOptimizer uses the formulation just before Section 2.1 of the Kingma and Ba paper rather than the formulation in Algorithm 1, the "epsilon" referred to here is "epsilon hat" in the paper.

The sparse implementation of this algorithm (used when the gradient is an IndexedSlices object, typically because of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) or an embedding lookup in the forward pass) does apply momentum to variable slices even if they were not used in the forward pass (meaning they have a gradient equal to zero). Momentum decay (beta1) is also applied to the entire momentum accumulator. This means that the sparse behavior is equivalent to the dense behavior (in contrast to some momentum implementations which ignore momentum unless a variable slice was actually used).

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **beta\_1**: A float value or a constant float tensor. The exponential decay rate for the 1st moment estimates.
* **beta\_2**: A float value or a constant float tensor. The exponential decay rate for the 2nd moment estimates.
* **epsilon**: A small constant for numerical stability. This epsilon is "epsilon hat" in the Kingma and Ba paper (in the formula just before Section 2.1), not the epsilon in Algorithm 1 of the paper.
* **amsgrad**: boolean. Whether to apply AMSGrad variant of this algorithm from the paper "On the Convergence of Adam and beyond".
* **name**: Optional name for the operations created when applying gradients. Defaults to "Adam". @compatibility(eager) When eager execution is enabled, learning\_rate, beta\_1, beta\_2, and epsilon can each be a callable that takes no **Arguments** and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions. @end\_compatibility
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.Adamax

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adamax#top_of_page)
* [Class Adamax](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adamax#class_adamax)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adamax#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adamax#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adamax#properties)

## Class Adamax

Optimizer that implements the Adamax algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.Adamax
* Class tf.compat.v2.keras.optimizers.Adamax
* Class tf.compat.v2.optimizers.Adamax
* Class tf.keras.optimizers.Adamax
* Class tf.optimizers.Adamax

Defined in [python/keras/optimizer\_v2/adamax.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/adamax.py).

It is a variant of Adam based on the infinity norm. Default parameters follow those provided in the paper. Adamax is sometimes superior to adam, specially in models with embeddings.

References see Section 7 of [Kingma et al., 2014](http://arxiv.org/abs/1412.6980) ([pdf](http://arxiv.org/pdf/1412.6980.pdf)).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    beta\_1=0.9,  
    beta\_2=0.999,  
    epsilon=1e-07,  
    name='Adamax',  
    \*\*kwargs  
)

Construct a new Adamax optimizer.

#### Initialization:

m\_0 <- 0 (Initialize initial 1st moment vector)  
v\_0 <- 0 (Initialize the exponentially weighted infinity norm)  
t <- 0 (Initialize timestep)

The update rule for variable with gradient g uses an optimization described at the end of section 7.1 of the paper:

t <- t + 1  
  
m\_t <- beta1 \* m\_{t-1} + (1 - beta1) \* g  
v\_t <- max(beta2 \* v\_{t-1}, abs(g))  
variable <- variable - learning\_rate / (1 - beta1^t) \* m\_t / (v\_t + epsilon)

Similar to AdamOptimizer, the epsilon is added for numerical stability (especially to get rid of division by zero when v\_t = 0).

Contrast to AdamOptimizer, the sparse implementation of this algorithm (used when the gradient is an IndexedSlices object, typically because of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) or an embedding lookup in the forward pass) only updates variable slices and corresponding m\_t, v\_t terms when that part of the variable was used in the forward pass. This means that the sparse behavior is contrast to the dense behavior (similar to some momentum implementations which ignore momentum unless a variable slice was actually used).

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **beta\_1**: A float value or a constant float tensor. The exponential decay rate for the 1st moment estimates.
* **beta\_2**: A float value or a constant float tensor. The exponential decay rate for the exponentially weighted infinity norm.
* **epsilon**: A small constant for numerical stability.
* **name**: Optional name for the operations created when applying gradients. Defaults to "Adamax".
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/deserialize#aliases)

Inverse of the serialize function.

### Aliases:

* tf.compat.v1.keras.optimizers.deserialize
* tf.compat.v2.keras.optimizers.deserialize
* tf.compat.v2.optimizers.deserialize
* tf.keras.optimizers.deserialize
* tf.optimizers.deserialize

tf.keras.optimizers.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/optimizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizers.py).

#### Arguments:

* **config**: Optimizer configuration dictionary.
* **custom\_objects**: Optional dictionary mapping names (strings) to custom objects (classes and functions) to be considered during deserialization.

#### Returns:

A Keras Optimizer instance.

# tf.keras.optimizers.Ftrl

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Ftrl#top_of_page)
* [Class Ftrl](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Ftrl#class_ftrl)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Ftrl#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Ftrl#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Ftrl#properties)

## Class Ftrl

Optimizer that implements the FTRL algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.Ftrl
* Class tf.compat.v2.keras.optimizers.Ftrl
* Class tf.compat.v2.optimizers.Ftrl
* Class tf.keras.optimizers.Ftrl
* Class tf.optimizers.Ftrl

Defined in [python/keras/optimizer\_v2/ftrl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/ftrl.py).

See Algorithm 1 of this [paper](https://www.eecs.tufts.edu/~dsculley/papers/ad-click-prediction.pdf). This version has support for both online L2 (the L2 penalty given in the paper above) and shrinkage-type L2 (which is the addition of an L2 penalty to the loss function).

#### Initialization:

t=0

n0=0

σ0=0

z0=0

Update (

i

is variable index):

t=t+1

nt,i=nt−1,i+gt,i2

σt,i=(nt,i−nt−1,i)/α

zt,i=zt−1,i+gt,i−σt,i∗wt,i

wt,i=−((β+n+t)/α+λ2)−1∗(zi−sgn(zi)∗λ1)if\abszi>λielse0

Check the documentation for the l2\_shrinkage\_regularization\_strength parameter for more details when shrinkage is enabled, where gradient is replaced with gradient\_with\_shrinkage.

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    learning\_rate\_power=-0.5,  
    initial\_accumulator\_value=0.1,  
    l1\_regularization\_strength=0.0,  
    l2\_regularization\_strength=0.0,  
    name='Ftrl',  
    l2\_shrinkage\_regularization\_strength=0.0,  
    \*\*kwargs  
)

Construct a new FTRL optimizer.

#### Args:

* **learning\_rate**: A float value or a constant float Tensor.
* **learning\_rate\_power**: A float value, must be less or equal to zero. Controls how the learning rate decreases during training. Use zero for a fixed learning rate.
* **initial\_accumulator\_value**: The starting value for accumulators. Only zero or positive values are allowed.
* **l1\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **l2\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Ftrl".
* **l2\_shrinkage\_regularization\_strength**: A float value, must be greater than or equal to zero. This differs from L2 above in that the L2 above is a stabilization penalty, whereas this L2 shrinkage is a magnitude penalty. The FTRL formulation can be written as: w\_{t+1} = argminw(\hat{g}{1:t}w + L1||w||\_1 + L2||w||\_2^2), where \hat{g} = g + (2L2\_shrinkagew), and g is the gradient of the loss function w.r.t. the weights w. Specifically, in the absence of L1 regularization, it is equivalent to the following update rule: w\_{t+1} = w\_t - lr\_t / (1 + 2L2lr\_t) \* g\_t - 2L2\_shrinkagelr\_t / (1 + 2L2lr\_t) \* w\_t where lr\_t is the learning rate at t. When input is sparse shrinkage will only happen on the active weights.\
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

#### Raises:

* **ValueError**: If one of the **Arguments** is invalid.

References See [paper](https://www.eecs.tufts.edu/~dsculley/papers/ad-click-prediction.pdf)

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/get#aliases)

Retrieves a Keras Optimizer instance.

### Aliases:

* tf.compat.v1.keras.optimizers.get
* tf.compat.v2.keras.optimizers.get
* tf.compat.v2.optimizers.get
* tf.keras.optimizers.get
* tf.optimizers.get

tf.keras.optimizers.get(identifier)

Defined in [python/keras/optimizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizers.py).

#### Arguments:

* **identifier**: Optimizer identifier, one of
  + String: name of an optimizer
  + Dictionary: configuration dictionary. - Keras Optimizer instance (it will be returned unchanged). - TensorFlow Optimizer instance (it will be wrapped as a Keras Optimizer).

#### Returns:

A Keras Optimizer instance.

#### Raises:

* **ValueError**: If identifier cannot be interpreted.

# tf.keras.optimizers.Nadam

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Nadam#top_of_page)
* [Class Nadam](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Nadam#class_nadam)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Nadam#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Nadam#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Nadam#properties)

## Class Nadam

Optimizer that implements the NAdam algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.Nadam
* Class tf.compat.v2.keras.optimizers.Nadam
* Class tf.compat.v2.optimizers.Nadam
* Class tf.keras.optimizers.Nadam
* Class tf.optimizers.Nadam

Defined in [python/keras/optimizer\_v2/nadam.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/nadam.py).

Much like Adam is essentially RMSprop with momentum, Nadam is Adam with Nesterov momentum.

#### Initialization:

m0:=0(Initialize 1st moment vector)

v0:=0(Initialize 2nd moment vector)

mu0:=1

t:=0(Initialize timestep)

#### Computes:

t:=t+1

μt:=β1∗(1−0.5∗0.960.004∗t)

g′:=g/(1−∏i=1tμi)

mt:=β1∗mt−1+(1−β1)∗g

m′:=mt/(1−∏i=1t+1μi)

vt:=β2∗vt−1+(1−β2)∗g∗g

v′:=vt/(1−β2t)

m¯:=(1−μt)∗g′+μt+1∗m′

θt:=θt−1−lr∗m¯/(v′+ϵ)

gradient is evaluated at theta(t) + momentum \* v(t), and the variables always store theta + beta\_1 \* m / sqrt(v) instead of theta.

References See [Dozat, T., 2015](http://cs229.stanford.edu/proj2015/054_report.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    beta\_1=0.9,  
    beta\_2=0.999,  
    epsilon=1e-07,  
    name='Nadam',  
    \*\*kwargs  
)

Construct a new Nadam optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **beta\_1**: A float value or a constant float tensor. The exponential decay rate for the 1st moment estimates.
* **beta\_2**: A float value or a constant float tensor. The exponential decay rate for the exponentially weighted infinity norm.
* **epsilon**: A small constant for numerical stability.
* **name**: Optional name for the operations created when applying gradients. Defaults to "Adamax".
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.Optimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer#top_of_page)
* [Class Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer#class_optimizer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer#aliases)
  + [Usage](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer#usage)
  + [Custom training loop with Keras models](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer#custom_training_loop_with_keras_models)

## Class Optimizer

Updated base class for optimizers.

### Aliases:

* Class tf.compat.v1.keras.optimizers.Optimizer
* Class tf.compat.v2.keras.optimizers.Optimizer
* Class tf.compat.v2.optimizers.Optimizer
* Class tf.keras.optimizers.Optimizer
* Class tf.optimizers.Optimizer

Defined in [python/keras/optimizer\_v2/optimizer\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/optimizer_v2.py).

This class defines the API to add Ops to train a model. You never use this class directly, but instead instantiate one of its subclasses such as [tf.keras.optimizers.SGD](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD), [tf.keras.optimizers.Adam](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Adam).

### Usage

# Create an optimizer with the desired parameters.  
opt = tf.keras.optimizers.SGD(learning\_rate=0.1)  
# `loss` is a callable that takes no argument and returns the value  
# to minimize.  
loss = lambda: 3 \* var1 \* var1 + 2 \* var2 \* var2  
# In graph mode, returns op that minimizes the loss by updating the listed  
# variables.  
opt\_op = opt.minimize(loss, var\_list=[var1, var2])  
opt\_op.run()  
# In eager mode, simply call minimize to update the list of variables.  
opt.minimize(loss, var\_list=[var1, var2])

### Custom training loop with Keras models

In Keras models, sometimes variables are created when the model is first called, instead of construction time. Examples include 1) sequential models without input shape pre-defined, or 2) subclassed models. Pass var\_list as callable in these cases.

#### Example:

opt = tf.keras.optimizers.SGD(learning\_rate=0.1)  
model = tf.keras.Sequential()  
model.add(tf.keras.layers.Dense(num\_hidden, activation='relu'))  
model.add(tf.keras.layers.Dense(num\_classes, activation='sigmoid')  
loss\_fn = lambda: tf.keras.losses.mse(model(input), output)  
var\_list\_fn = lambda: model.trainable\_weights  
for input, output in data:  
  opt.minimize(loss\_fn, var\_list\_fn)

### Processing gradients before applying them.

Calling minimize() takes care of both computing the gradients and applying them to the variables. If you want to process the gradients before applying them you can instead use the optimizer in three steps:

1. Compute the gradients with [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape).
2. Process the gradients as you wish.
3. Apply the processed gradients with apply\_gradients().

#### Example:

# Create an optimizer.  
opt = tf.keras.optimizers.SGD(learning\_rate=0.1)  
  
# Compute the gradients for a list of variables.  
with tf.GradientTape() as tape:  
  loss = <call\_loss\_function>  
vars = <list\_of\_variables>  
grads = tape.gradient(loss, vars)  
processed\_grads = [process\_gradient(g) for g in grads]  
grads\_and\_vars = zip(processed\_grads, var\_list)  
  
# grads\_and\_vars is a list of tuples (gradient, variable).  Do whatever you  
# need to the 'gradient' part, for example cap them, etc.  
capped\_grads\_and\_vars = [(MyCapper(gv[0]), gv[1]) for gv in grads\_and\_vars]  
  
# Ask the optimizer to apply the capped gradients.  
opt.apply\_gradients(capped\_grads\_and\_vars)

### Use with tf.distribute.Strategy.

This optimizer class is [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy) aware, which means it automatically sums gradients across all replicas. To average gradients, you divide your loss by the global batch size, which is done automatically if you use [tf.keras](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras) built-in training or evaluation loops. See the reductionargument of your loss which should be set to [tf.keras.losses.Reduction.SUM\_OVER\_BATCH\_SIZE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#SUM_OVER_BATCH_SIZE)for averaging or [tf.keras.losses.Reduction.SUM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/losses/Reduction#SUM) for not.

If you are not using these and you want to average gradients, you should use [tf.math.reduce\_sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum) to add up your per-example losses and then divide by the global batch size. Note that when using [tf.distribute.Strategy](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Strategy), the first component of a tensor's shape is the replica-local batch size, which is off by a factor equal to the number of replicas being used to compute a single step. As a result, using [tf.math.reduce\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean) will give the wrong answer, resulting in gradients that can be many times too big.

### Variable Constraint

All Keras optimizers respect variable constraints. If constraint function is passed to any variable, the constraint will be applied to the variable after the gradient has been applied to the variable. Important: If gradient is sparse tensor, variable constraint is not supported.

### Thread Compatibility

The entire optimizer is currently thread compatible, not thread-safe. The user needs to perform synchronization if necessary.

### Slots

Many optimizer subclasses, such as Adam and Adagrad allocate and manage additional variables associated with the variables to train. These are called *Slots*. Slots have names and you can ask the optimizer for the names of the slots that it uses. Once you have a slot name you can ask the optimizer for the variable it created to hold the slot value.

This can be useful if you want to log debug a training algorithm, report stats about the slots, etc.

### Hyper parameters

These are **Arguments** passed to the optimizer subclass constructor (the \_\_init\_\_ method), and then passed to self.\_set\_hyper(). They can be either regular Python values (like 1.0), tensors, or callables. If they are callable, the callable will be called during apply\_gradients() to get the value for the hyper parameter.

Hyper parameters can be overwritten through user code:

#### Example:

# Create an optimizer with the desired parameters.  
opt = tf.keras.optimizers.SGD(learning\_rate=0.1)  
# `loss` is a callable that takes no argument and returns the value  
# to minimize.  
loss = lambda: 3 \* var1 + 2 \* var2  
# In eager mode, simply call minimize to update the list of variables.  
opt.minimize(loss, var\_list=[var1, var2])  
# update learning rate  
opt.learning\_rate = 0.05  
opt.minimize(loss, var\_list=[var1, var2])

### Write a customized optimizer.

If you intend to create your own optimization algorithm, simply inherit from this class and override the following methods:

* resource\_apply\_dense (update variable given gradient tensor is dense)
* resource\_apply\_sparse (update variable given gradient tensor is sparse)
* create\_slots (if your optimizer algorithm requires additional variables)
* get\_config (serialization of the optimizer, include all hyper parameters)

## \_\_init\_\_

\_\_init\_\_(  
    name,  
    \*\*kwargs  
)

Create a new Optimizer.

This must be called by the constructors of subclasses. Note that Optimizer instances should not bind to a single graph, and so shouldn't keep Tensors as member variables. Generally you should be able to use the \_set\_hyper()/state.get\_hyper() facility instead.

This class in stateful and thread-compatible.

#### Args:

* **name**: A non-empty string. The name to use for accumulators created for the optimizer.
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

#### Raises:

* **ValueError**: If name is malformed.
* **RuntimeError**: If \_create\_slots has been overridden instead of \_create\_vars.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

@classmethod  
from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

Returns the config of the optimimizer.

An optimizer config is a Python dictionary (serializable) containing the configuration of an optimizer. The same optimizer can be reinstantiated later (without any saved state) from this configuration.

#### Returns:

Python dictionary.

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

# tf.keras.optimizers.RMSprop

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/RMSprop#top_of_page)
* [Class RMSprop](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/RMSprop#class_rmsprop)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/RMSprop#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/RMSprop#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/RMSprop#used_in_the_tutorials)

## Class RMSprop

Optimizer that implements the RMSprop algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.RMSprop
* Class tf.compat.v2.keras.optimizers.RMSprop
* Class tf.compat.v2.optimizers.RMSprop
* Class tf.keras.optimizers.RMSprop
* Class tf.optimizers.RMSprop

Defined in [python/keras/optimizer\_v2/rmsprop.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/rmsprop.py).

### Used in the guide:

* [Keras custom callbacks](https://www.tensorflow.org/beta/guide/keras/custom_callback)
* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Saving and Serializing Models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/saving_and_serializing)
* [The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Regression: Predict fuel efficiency](https://www.tensorflow.org/beta/tutorials/keras/basic_regression)
* [Transfer Learning Using Pretrained ConvNets](https://www.tensorflow.org/beta/tutorials/images/transfer_learning)

A detailed description of rmsprop.

* maintain a moving (discounted) average of the square of gradients
* divide gradient by the root of this average

meansquaret=rho∗meansquaret−1+(1−rho)∗gradient∗∗2

momt=momentum∗momt−1+learningrate∗gradient//meansquaret+ϵ

variablet:=variablet−1−momt

This implementation of RMSprop uses plain momentum, not Nesterov momentum.

The centered version additionally maintains a moving average of the gradients, and uses that average to estimate the variance:

meangradt=rho∗meangradt−1+(1−rho)∗gradient

meansquaret=rho∗meansquaret−1+(1−rho)∗gradient∗∗2

momt=momentum∗momt−1+learningrate∗gradient/sqrt(meansquaret−meangradt∗∗2+epsilon)

variablet:=variablet−1−momt

References See ([pdf] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\_slides\_lec6.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    rho=0.9,  
    momentum=0.0,  
    epsilon=1e-07,  
    centered=False,  
    name='RMSprop',  
    \*\*kwargs  
)

Construct a new RMSprop optimizer.

Note that in the dense implementation of this algorithm, variables and their corresponding accumulators (momentum, gradient moving average, square gradient moving average) will be updated even if the gradient is zero (i.e. accumulators will decay, momentum will be applied). The sparse implementation (used when the gradient is an IndexedSlices object, typically because of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) or an embedding lookup in the forward pass) will not update variable slices or their accumulators unless those slices were used in the forward pass (nor is there an "eventual" correction to account for these omitted updates). This leads to more efficient updates for large embedding lookup tables (where most of the slices are not accessed in a particular graph execution), but differs from the published algorithm.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **rho**: Discounting factor for the history/coming gradient
* **momentum**: A scalar tensor.
* **epsilon**: Small value to avoid zero denominator.
* **centered**: If True, gradients are normalized by the estimated variance of the gradient; if False, by the uncentered second moment. Setting this to True may help with training, but is slightly more expensive in terms of computation and memory. Defaults to False.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "RMSprop". @compatibility(eager) When eager execution is enabled, learning\_rate, decay, momentum, and epsilon can each be a callable that takes no **Arguments** and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions. @end\_compatibility
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created

tf.keras.optimizers.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/serialize#aliases)

Aliases:

* tf.compat.v1.keras.optimizers.serialize
* tf.compat.v2.keras.optimizers.serialize
* tf.compat.v2.optimizers.serialize
* tf.keras.optimizers.serialize
* tf.optimizers.serialize

tf.keras.optimizers.serialize(optimizer)

Defined in [python/keras/optimizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizers.py).

# tf.keras.optimizers.SGD

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD#top_of_page)
* [Class SGD](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD#class_sgd)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/SGD#used_in_the_tutorials)

## Class SGD

Stochastic gradient descent and momentum optimizer.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer)

### Aliases:

* Class tf.compat.v1.keras.optimizers.SGD
* Class tf.compat.v2.keras.optimizers.SGD
* Class tf.compat.v2.optimizers.SGD
* Class tf.keras.optimizers.SGD
* Class tf.optimizers.SGD

Defined in [python/keras/optimizer\_v2/gradient\_descent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/gradient_descent.py).

### Used in the guide:

* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Using GPUs](https://www.tensorflow.org/beta/guide/using_gpu)
* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

### Used in the tutorials:

* [Multi-worker Training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_keras)

#### Computes:

theta(t+1) = theta(t) - learning\_rate \* gradient  
gradient is evaluated at theta(t).

or Computes (if nesterov = False):

v(t+1) = momentum \* v(t) - learning\_rate \* gradient  
theta(t+1) = theta(t) + v(t+1)  
if `nesterov` is False, gradient is evaluated at theta(t).  
if `nesterov` is True, gradient is evaluated at theta(t) + momentum \* v(t),  
  and the variables always store theta + m v instead of theta

Some of the args below are hyperparameters, where a hyperparameter is defined as a scalar Tensor, a regular Python value, or a callable (which will be evaluated when apply\_gradients is called) returning a scalar Tensor or a Python value.

# References

nesterov = True, See [Sutskever et al., 2013](  
  http://jmlr.org/proceedings/papers/v28/sutskever13.pdf).

#### Eager Compatibility

When eager execution is enabled, learning\_rate can be a callable that takes no **Arguments** and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.01,  
    momentum=0.0,  
    nesterov=False,  
    name='SGD',  
    \*\*kwargs  
)

Construct a new Stochastic Gradient Descent or Momentum optimizer.

#### Arguments:

* **learning\_rate**: float hyperparameter >= 0. Learning rate.
* **momentum**: float hyperparameter >= 0 that accelerates SGD in the relevant direction and dampens oscillations.
* **nesterov**: boolean. Whether to apply Nesterov momentum.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to 'SGD'.
* **\*\*kwargs**: keyword **Arguments**. Allowed to be {clipnorm, clipvalue, lr, decay}. clipnorm is clip gradients by norm; clipvalue is clip gradients by value, decay is included for backward compatibility to allow time inverse decay of learning rate. lr is included for backward compatibility, recommended to use learning\_rate instead.

## Properties

### iterations

Variable. The number of training steps this Optimizer has run.

### weights

Returns variables of this Optimizer based on the order created.

## Methods

### add\_slot

add\_slot(  
    var,  
    slot\_name,  
    initializer='zeros'  
)

Add a new slot variable for var.

### add\_weight

add\_weight(  
    name,  
    shape,  
    dtype=None,  
    initializer='zeros',  
    trainable=None,  
    synchronization=tf.VariableSynchronization.AUTO,  
    aggregation=tf.compat.v1.VariableAggregation.NONE  
)

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.

### from\_config

from\_config(  
    cls,  
    config,  
    custom\_objects=None  
)

Creates an optimizer from its config.

This method is the reverse of get\_config, capable of instantiating the same optimizer from the config dictionary.

#### Arguments:

* **config**: A Python dictionary, typically the output of get\_config.
* **custom\_objects**: A Python dictionary mapping names to additional Python objects used to create this optimizer, such as a function used for a hyperparameter.

#### Returns:

An optimizer instance.

### get\_config

get\_config()

### get\_gradients

get\_gradients(  
    loss,  
    params  
)

Returns gradients of loss with respect to params.

#### Arguments:

* **loss**: Loss tensor.
* **params**: List of variables.

#### Returns:

List of gradient tensors.

#### Raises:

* **ValueError**: In case any gradient cannot be computed (e.g. if gradient function not implemented).

### get\_slot

get\_slot(  
    var,  
    slot\_name  
)

### get\_slot\_names

get\_slot\_names()

A list of names for this optimizer's slots.

### get\_updates

get\_updates(  
    loss,  
    params  
)

### get\_weights

get\_weights()

### minimize

minimize(  
    loss,  
    var\_list,  
    grad\_loss=None,  
    name=None  
)

Minimize loss by updating var\_list.

This method simply computes gradient using [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and calls apply\_gradients(). If you want to process the gradient before applying then call [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) and apply\_gradients() explicitly instead of using this function.

#### Args:

* **loss**: A callable taking no **Arguments** which returns the value to minimize.
* **var\_list**: list or tuple of Variable objects to update to minimize loss, or a callable returning the list or tuple of Variable objects. Use callable when the variable list would otherwise be incomplete before minimize since the variables are created at the first time loss is called.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.
* **name**: Optional name for the returned operation.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

### set\_weights

set\_weights(weights)

### variables

variables()

Returns variables of this Optimizer based on the order created.

Module: tf.keras.optimizers.schedules

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules#functions)

Public API for tf.keras.optimizers.schedules namespace.

Aliases:

* Module tf.compat.v2.keras.optimizers.schedules
* Module tf.compat.v2.optimizers.schedules
* Module tf.keras.optimizers.schedules
* Module tf.optimizers.schedules

Defined in [python/keras/api/\_v2/keras/optimizers/schedules/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/optimizers/schedules/__init__.py).

Classes

[class ExponentialDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay): A LearningRateSchedule that uses an exponential decay schedule.

[class InverseTimeDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/InverseTimeDecay): A LearningRateSchedule that uses an inverse time decay schedule.

[class LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule): A serializable learning rate decay schedule.

[class PiecewiseConstantDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PiecewiseConstantDecay): A LearningRateSchedule that uses a piecewise constant decay schedule.

[class PolynomialDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay): A LearningRateSchedule that uses a polynomial decay schedule.

Functions

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize)

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize)

tf.keras.optimizers.schedules.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize#aliases)

Aliases:

* tf.compat.v1.keras.optimizers.schedules.deserialize
* tf.compat.v2.keras.optimizers.schedules.deserialize
* tf.compat.v2.optimizers.schedules.deserialize
* tf.keras.optimizers.schedules.deserialize
* tf.optimizers.schedules.deserialize

tf.keras.optimizers.schedules.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

# tf.keras.optimizers.schedules.ExponentialDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay#top_of_page)
* [Class ExponentialDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay#class_exponentialdecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/ExponentialDecay#__init__)

## Class ExponentialDecay

A LearningRateSchedule that uses an exponential decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.optimizers.schedules.ExponentialDecay
* Class tf.compat.v2.keras.optimizers.schedules.ExponentialDecay
* Class tf.compat.v2.optimizers.schedules.ExponentialDecay
* Class tf.keras.optimizers.schedules.ExponentialDecay
* Class tf.optimizers.schedules.ExponentialDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    decay\_steps,  
    decay\_rate,  
    staircase=False,  
    name=None  
)

Applies exponential decay to the learning rate.

When training a model, it is often recommended to lower the learning rate as the training progresses. This schedule applies an exponential decay function to an optimizer step, given a provided initial learning rate.

The schedule a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions. It is computed as:

def decayed\_learning\_rate(step):  
  return initial\_learning\_rate \* decay\_rate ^ (step / decay\_steps)

If the argument staircase is True, then step / decay\_steps is an integer division and the decayed learning rate follows a staircase function.

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. Example: When fitting a Keras model, decay every 100000 steps with a base of 0.96:

initial\_learning\_rate = 0.1  
lr\_schedule = tf.keras.optimizers.schedules.ExponentialDecay(  
    initial\_learning\_rate,  
    decay\_steps=100000,  
    decay\_rate=0.96,  
    staircase=True)  
  
model.compile(optimizer=tf.keras.optimizers.SGD(learning\_rate=lr\_schedule),  
              loss='sparse\_categorical\_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(data, labels, epochs=5)

The learning rate schedule is also serializable and deserializable using[tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Must be positive. See the decay computation above.
* **decay\_rate**: A scalar float32 or float64 Tensor or a Python number. The decay rate.
* **staircase**: Boolean. If True decay the learning rate at discrete intervals
* **name**: String. Optional name of the operation. Defaults to 'ExponentialDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.optimizers.schedules.InverseTimeDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/InverseTimeDecay#top_of_page)
* [Class InverseTimeDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/InverseTimeDecay#class_inversetimedecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/InverseTimeDecay#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/InverseTimeDecay#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/InverseTimeDecay#methods)

## Class InverseTimeDecay

A LearningRateSchedule that uses an inverse time decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.optimizers.schedules.InverseTimeDecay
* Class tf.compat.v2.keras.optimizers.schedules.InverseTimeDecay
* Class tf.compat.v2.optimizers.schedules.InverseTimeDecay
* Class tf.keras.optimizers.schedules.InverseTimeDecay
* Class tf.optimizers.schedules.InverseTimeDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    decay\_steps,  
    decay\_rate,  
    staircase=False,  
    name=None  
)

Applies inverse time decay to the initial learning rate.

When training a model, it is often recommended to lower the learning rate as the training progresses. This schedule applies the inverse decay function to an optimizer step, given a provided initial learning rate. It requires a step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The schedule a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions. It is computed as:

def decayed\_learning\_rate(step):  
  return initial\_learning\_rate / (1 + decay\_rate \* step / decay\_step)

or, if staircase is True, as:

def decayed\_learning\_rate(step):  
  return initial\_learning\_rate / (1 + decay\_rate \* floor(step / decay\_step))

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. Example: Fit a Keras model when decaying 1/t with a rate of 0.5:

...  
initial\_learning\_rate = 0.1  
decay\_steps = 1.0  
decay\_rate = 0.5  
learning\_rate\_fn = keras.optimizers.schedules.InverseTimeDecay(  
  initial\_learning\_rate, global\_step, decay\_steps, decay\_rate)  
  
model.compile(optimizer=tf.keras.optimizers.SGD(  
                  learning\_rate=learning\_rate\_fn),  
              loss='sparse\_categorical\_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(data, labels, epochs=5)

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **decay\_steps**: How often to apply decay.
* **decay\_rate**: A Python number. The decay rate.
* **staircase**: Whether to apply decay in a discrete staircase, as opposed to continuous, fashion.
* **name**: String. Optional name of the operation. Defaults to 'InverseTimeDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.optimizers.schedules.LearningRateSchedule

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#top_of_page)
* [Class LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#class_learningrateschedule)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#__call__)
  + [from\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#from_config)
  + [get\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule#get_config)

## Class LearningRateSchedule

A serializable learning rate decay schedule.

### Aliases:

* Class tf.compat.v1.keras.optimizers.schedules.LearningRateSchedule
* Class tf.compat.v2.keras.optimizers.schedules.LearningRateSchedule
* Class tf.compat.v2.optimizers.schedules.LearningRateSchedule
* Class tf.keras.optimizers.schedules.LearningRateSchedule
* Class tf.optimizers.schedules.LearningRateSchedule

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

LearningRateSchedules can be passed in as the learning rate of optimizers in[tf.keras.optimizers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers). They can be serialized and deserialized using[tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.optimizers.schedules.PiecewiseConstantDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PiecewiseConstantDecay#top_of_page)
* [Class PiecewiseConstantDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PiecewiseConstantDecay#class_piecewiseconstantdecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PiecewiseConstantDecay#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PiecewiseConstantDecay#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PiecewiseConstantDecay#methods)

## Class PiecewiseConstantDecay

A LearningRateSchedule that uses a piecewise constant decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.optimizers.schedules.PiecewiseConstantDecay
* Class tf.compat.v2.keras.optimizers.schedules.PiecewiseConstantDecay
* Class tf.compat.v2.optimizers.schedules.PiecewiseConstantDecay
* Class tf.keras.optimizers.schedules.PiecewiseConstantDecay
* Class tf.optimizers.schedules.PiecewiseConstantDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    boundaries,  
    values,  
    name=None  
)

Piecewise constant from boundaries and interval values.

The function returns a 1-arg callable to compute the piecewise constant when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions.

Example: use a learning rate that's 1.0 for the first 100001 steps, 0.5 for the next 10000 steps, and 0.1 for any additional steps.

step = tf.Variable(0, trainable=False)  
boundaries = [100000, 110000]  
values = [1.0, 0.5, 0.1]  
learning\_rate\_fn = keras.optimizers.schedules.PiecewiseConstantDecay(  
    boundaries, values)  
  
# Later, whenever we perform an optimization step, we pass in the step.  
learning\_rate = learning\_rate\_fn(step)

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. The learning rate schedule is also serializable and deserializable using [tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **boundaries**: A list of Tensors or ints or floats with strictly increasing entries, and with all elements having the same type as the optimizer step.
* **values**: A list of Tensors or floats or ints that specifies the values for the intervals defined by boundaries. It should have one more element than boundaries, and all elements should have the same type.
* **name**: A string. Optional name of the operation. Defaults to 'PiecewiseConstant'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as the boundary tensors.

The output of the 1-arg function that takes the step is values[0] when step <= boundaries[0],values[1] when step > boundaries[0] and step <= boundaries[1], ..., and values[-1] when step > boundaries[-1].

#### Raises:

* **ValueError**: if the number of elements in the lists do not match.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

# tf.keras.optimizers.schedules.PolynomialDecay

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay#top_of_page)
* [Class PolynomialDecay](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay#class_polynomialdecay)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/PolynomialDecay#methods)

## Class PolynomialDecay

A LearningRateSchedule that uses a polynomial decay schedule.

Inherits From: [LearningRateSchedule](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/LearningRateSchedule)

### Aliases:

* Class tf.compat.v1.keras.optimizers.schedules.PolynomialDecay
* Class tf.compat.v2.keras.optimizers.schedules.PolynomialDecay
* Class tf.compat.v2.optimizers.schedules.PolynomialDecay
* Class tf.keras.optimizers.schedules.PolynomialDecay
* Class tf.optimizers.schedules.PolynomialDecay

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

## \_\_init\_\_

\_\_init\_\_(  
    initial\_learning\_rate,  
    decay\_steps,  
    end\_learning\_rate=0.0001,  
    power=1.0,  
    cycle=False,  
    name=None  
)

Applies a polynomial decay to the learning rate.

It is commonly observed that a monotonically decreasing learning rate, whose degree of change is carefully chosen, results in a better performing model. This schedule applies a polynomial decay function to an optimizer step, given a provided initial\_learning\_rate, to reach an end\_learning\_rate in the given decay\_steps.

It requires a step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The schedule is a 1-arg callable that produces a decayed learning rate when passed the current optimizer step. This can be useful for changing the learning rate value across different invocations of optimizer functions. It is computed as:

def decayed\_learning\_rate(step):  
  step = min(step, decay\_steps)  
  return ((initial\_learning\_rate - end\_learning\_rate) \*  
          (1 - step / decay\_steps) ^ (power)  
         ) + end\_learning\_rate

If cycle is True then a multiple of decay\_steps is used, the first one that is bigger than step.

def decayed\_learning\_rate(step):  
  decay\_steps = decay\_steps \* ceil(step / decay\_steps)  
  return ((initial\_learning\_rate - end\_learning\_rate) \*  
          (1 - step / decay\_steps) ^ (power)  
         ) + end\_learning\_rate

You can pass this schedule directly into a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) as the learning rate. Example: Fit a model while decaying from 0.1 to 0.01 in 10000 steps using sqrt (i.e. power=0.5):

...  
starter\_learning\_rate = 0.1  
end\_learning\_rate = 0.01  
decay\_steps = 10000  
learning\_rate\_fn = tf.keras.optimizers.schedules.PolynomialDecay(  
    starter\_learning\_rate,  
    decay\_steps,  
    end\_learning\_rate,  
    power=0.5)  
  
model.compile(optimizer=tf.keras.optimizers.SGD(  
                  learning\_rate=learning\_rate\_fn),  
              loss='sparse\_categorical\_crossentropy',  
              metrics=['accuracy'])  
  
model.fit(data, labels, epochs=5)

The learning rate schedule is also serializable and deserializable using[tf.keras.optimizers.schedules.serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize) and[tf.keras.optimizers.schedules.deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/deserialize).

#### Args:

* **initial\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Must be positive. See the decay computation above.
* **end\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The minimal end learning rate.
* **power**: A scalar float32 or float64 Tensor or a Python number. The power of the polynomial. Defaults to linear, 1.0.
* **cycle**: A boolean, whether or not it should cycle beyond decay\_steps.
* **name**: String. Optional name of the operation. Defaults to 'PolynomialDecay'.

#### Returns:

A 1-arg callable learning rate schedule that takes the current optimizer step and outputs the decayed learning rate, a scalar Tensor of the same type as initial\_learning\_rate.

## Methods

### \_\_call\_\_

\_\_call\_\_(step)

### from\_config

from\_config(  
    cls,  
    config  
)

Instantiates a LearningRateSchedule from its config.

#### Args:

* **config**: Output of get\_config().

#### Returns:

A LearningRateSchedule instance.

### get\_config

get\_config()

tf.keras.optimizers.schedules.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/schedules/serialize#aliases)

Aliases:

* tf.compat.v1.keras.optimizers.schedules.serialize
* tf.compat.v2.keras.optimizers.schedules.serialize
* tf.compat.v2.optimizers.schedules.serialize
* tf.keras.optimizers.schedules.serialize
* tf.optimizers.schedules.serialize

tf.keras.optimizers.schedules.serialize(learning\_rate\_schedule)

Defined in [python/keras/optimizer\_v2/learning\_rate\_schedule.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/optimizer_v2/learning_rate_schedule.py).

Module: tf.keras.preprocessing

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing#aliases)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing#modules)

Keras data preprocessing utils.

Aliases:

* Module tf.compat.v2.keras.preprocessing
* Module tf.keras.preprocessing

Defined in [python/keras/api/\_v2/keras/preprocessing/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/preprocessing/__init__.py).

Modules

[image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image) module: Set of tools for real-time data augmentation on image data.

[sequence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence) module: Utilities for preprocessing sequence data.

[text](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text) module: Utilities for text input preprocessing.

Module: tf.keras.preprocessing.image

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image#functions)

Set of tools for real-time data augmentation on image data.

Aliases:

* Module tf.compat.v2.keras.preprocessing.image
* Module tf.keras.preprocessing.image

Defined in [python/keras/api/\_v2/keras/preprocessing/image/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/preprocessing/image/__init__.py).

Classes

[class DirectoryIterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/DirectoryIterator): Iterator capable of reading images from a directory on disk.

[class ImageDataGenerator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator): Generate batches of tensor image data with real-time data augmentation.

[class Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator)

[class NumpyArrayIterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/NumpyArrayIterator): Iterator yielding data from a Numpy array.

Functions

[apply\_affine\_transform(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_affine_transform): Applies an affine transformation specified by the parameters given.

[apply\_brightness\_shift(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_brightness_shift): Performs a brightness shift.

[apply\_channel\_shift(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_channel_shift): Performs a channel shift.

[array\_to\_img(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/array_to_img): Converts a 3D Numpy array to a PIL Image instance.

[img\_to\_array(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/img_to_array): Converts a PIL Image instance to a Numpy array.

[load\_img(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/load_img): Loads an image into PIL format.

[random\_brightness(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_brightness): Performs a random brightness shift.

[random\_channel\_shift(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_channel_shift): Performs a random channel shift.

[random\_rotation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_rotation): Performs a random rotation of a Numpy image tensor.

[random\_shear(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_shear): Performs a random spatial shear of a Numpy image tensor.

[random\_shift(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_shift): Performs a random spatial shift of a Numpy image tensor.

[random\_zoom(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_zoom): Performs a random spatial zoom of a Numpy image tensor.

[save\_img(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/save_img): Saves an image stored as a Numpy array to a path or file object.

tf.keras.preprocessing.image.apply\_affine\_transform

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_affine_transform#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_affine_transform#aliases)

Applies an affine transformation specified by the parameters given.

Aliases:

* tf.compat.v1.keras.preprocessing.image.apply\_affine\_transform
* tf.compat.v2.keras.preprocessing.image.apply\_affine\_transform
* tf.keras.preprocessing.image.apply\_affine\_transform

tf.keras.preprocessing.image.apply\_affine\_transform(  
    x,  
    theta=0,  
    tx=0,  
    ty=0,  
    shear=0,  
    zx=1,  
    zy=1,  
    row\_axis=0,  
    col\_axis=1,  
    channel\_axis=2,  
    fill\_mode='nearest',  
    cval=0.0,  
    order=1  
)

**Arguments**

x: 2D numpy array, single image.  
theta: Rotation angle in degrees.  
tx: Width shift.  
ty: Heigh shift.  
shear: Shear angle in degrees.  
zx: Zoom in x direction.  
zy: Zoom in y direction  
row\_axis: Index of axis for rows in the input image.  
col\_axis: Index of axis for columns in the input image.  
channel\_axis: Index of axis for channels in the input image.  
fill\_mode: Points outside the boundaries of the input  
    are filled according to the given mode  
    (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).  
cval: Value used for points outside the boundaries  
    of the input if `mode='constant'`.  
order: int, order of interpolation

**Returns**

The transformed version of the input.

tf.keras.preprocessing.image.apply\_brightness\_shift

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_brightness_shift#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_brightness_shift#aliases)

Performs a brightness shift.

Aliases:

* tf.compat.v1.keras.preprocessing.image.apply\_brightness\_shift
* tf.compat.v2.keras.preprocessing.image.apply\_brightness\_shift
* tf.keras.preprocessing.image.apply\_brightness\_shift

tf.keras.preprocessing.image.apply\_brightness\_shift(  
    x,  
    brightness  
)

**Arguments**

x: Input tensor. Must be 3D.  
brightness: Float. The new brightness value.  
channel\_axis: Index of axis for channels in the input tensor.

**Returns**

Numpy image tensor.

**Raises**

ValueError if `brightness\_range` isn't a tuple.

tf.keras.preprocessing.image.apply\_channel\_shift

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_channel_shift#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/apply_channel_shift#aliases)

Performs a channel shift.

Aliases:

* tf.compat.v1.keras.preprocessing.image.apply\_channel\_shift
* tf.compat.v2.keras.preprocessing.image.apply\_channel\_shift
* tf.keras.preprocessing.image.apply\_channel\_shift

tf.keras.preprocessing.image.apply\_channel\_shift(  
    x,  
    intensity,  
    channel\_axis=0  
)

**Arguments**

x: Input tensor. Must be 3D.  
intensity: Transformation intensity.  
channel\_axis: Index of axis for channels in the input tensor.

**Returns**

Numpy image tensor.

# tf.keras.preprocessing.image.array\_to\_img

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/array_to_img#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/array_to_img#aliases)

Converts a 3D Numpy array to a PIL Image instance.

### Aliases:

* tf.compat.v1.keras.preprocessing.image.array\_to\_img
* tf.compat.v2.keras.preprocessing.image.array\_to\_img
* tf.keras.preprocessing.image.array\_to\_img

tf.keras.preprocessing.image.array\_to\_img(  
    x,  
    data\_format=None,  
    scale=True,  
    dtype=None  
)

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

#### Arguments:

* **x**: Input Numpy array.
* **data\_format**: Image data format. either "channels\_first" or "channels\_last".
* **scale**: Whether to rescale image values to be within [0, 255].
* **dtype**: Dtype to use.

#### Returns:

A PIL Image instance.

#### Raises:

* **ImportError**: if PIL is not available.
* **ValueError**: if invalid x or data\_format is passed.

# tf.keras.preprocessing.image.DirectoryIterator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/DirectoryIterator#top_of_page)
* [Class DirectoryIterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/DirectoryIterator#class_directoryiterator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/DirectoryIterator#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/DirectoryIterator#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/DirectoryIterator#properties)

## Class DirectoryIterator

Iterator capable of reading images from a directory on disk.

Inherits From: [Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator)

### Aliases:

* Class tf.compat.v1.keras.preprocessing.image.DirectoryIterator
* Class tf.compat.v2.keras.preprocessing.image.DirectoryIterator
* Class tf.keras.preprocessing.image.DirectoryIterator

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

#### Arguments:

* **directory**: Path to the directory to read images from. Each subdirectory in this directory will be considered to contain images from one class, or alternatively you could specify class subdirectories via the classes argument.
* **image\_data\_generator**: Instance of ImageDataGenerator to use for random transformations and normalization.
* **target\_size**: tuple of integers, dimensions to resize input images to.
* **color\_mode**: One of "rgb", "rgba", "grayscale". Color mode to read images.
* **classes**: Optional list of strings, names of subdirectories containing images from each class (e.g. ["dogs", "cats"]). It will be computed automatically if not set.
* **class\_mode**: Mode for yielding the targets: "binary": binary targets (if there are only two classes), "categorical": categorical targets, "sparse": integer targets, "input": targets are images identical to input images (mainly used to work with autoencoders), None: no targets get yielded (only input images are yielded).
* **batch\_size**: Integer, size of a batch.
* **shuffle**: Boolean, whether to shuffle the data between epochs.
* **seed**: Random seed for data shuffling.
* **data\_format**: String, one of channels\_first, channels\_last.
* **save\_to\_dir**: Optional directory where to save the pictures being yielded, in a viewable format. This is useful for visualizing the random transformations being applied, for debugging purposes.
* **save\_prefix**: String prefix to use for saving sample images (if save\_to\_dir is set).
* **save\_format**: Format to use for saving sample images (if save\_to\_dir is set).
* **subset**: Subset of data ("training" or "validation") if validation\_split is set in ImageDataGenerator.
* **interpolation**: Interpolation method used to resample the image if the target size is different from that of the loaded image. Supported methods are "nearest", "bilinear", and "bicubic". If PIL version 1.1.3 or newer is installed, "lanczos" is also supported. If PIL version 3.4.0 or newer is installed, "box" and "hamming" are also supported. By default, "nearest" is used.
* **dtype**: Dtype to use for generated arrays.

## \_\_init\_\_

\_\_init\_\_(  
    directory,  
    image\_data\_generator,  
    target\_size=(256, 256),  
    color\_mode='rgb',  
    classes=None,  
    class\_mode='categorical',  
    batch\_size=32,  
    shuffle=True,  
    seed=None,  
    data\_format=None,  
    save\_to\_dir=None,  
    save\_prefix='',  
    save\_format='png',  
    follow\_links=False,  
    subset=None,  
    interpolation='nearest',  
    dtype=None  
)

## Properties

### filepaths

### labels

### sample\_weight

## Methods

### \_\_getitem\_\_

\_\_getitem\_\_(idx)

### \_\_iter\_\_

\_\_iter\_\_()

### \_\_len\_\_

\_\_len\_\_()

### next

next()

For python 2.x.

# Returns

The next batch.

### on\_epoch\_end

on\_epoch\_end()

### reset

reset()

### set\_processing\_attrs

set\_processing\_attrs(  
    image\_data\_generator,  
    target\_size,  
    color\_mode,  
    data\_format,  
    save\_to\_dir,  
    save\_prefix,  
    save\_format,  
    subset,  
    interpolation  
)

Sets attributes to use later for processing files into a batch.

# Arguments

image\_data\_generator: Instance of `ImageDataGenerator`  
    to use for random transformations and normalization.  
target\_size: tuple of integers, dimensions to resize input images to.  
color\_mode: One of `"rgb"`, `"rgba"`, `"grayscale"`.  
    Color mode to read images.  
data\_format: String, one of `channels\_first`, `channels\_last`.  
save\_to\_dir: Optional directory where to save the pictures  
    being yielded, in a viewable format. This is useful  
    for visualizing the random transformations being  
    applied, for debugging purposes.  
save\_prefix: String prefix to use for saving sample  
    images (if `save\_to\_dir` is set).  
save\_format: Format to use for saving sample images  
    (if `save\_to\_dir` is set).  
subset: Subset of data (`"training"` or `"validation"`) if  
    validation\_split is set in ImageDataGenerator.  
interpolation: Interpolation method used to resample the image if the  
    target size is different from that of the loaded image.  
    Supported methods are "nearest", "bilinear", and "bicubic".  
    If PIL version 1.1.3 or newer is installed, "lanczos" is also  
    supported. If PIL version 3.4.0 or newer is installed, "box" and  
    "hamming" are also supported. By default, "nearest" is used.

## Class Members

* allowed\_class\_modes
* white\_list\_formats

# tf.keras.preprocessing.image.ImageDataGenerator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#top_of_page)
* [Class ImageDataGenerator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#class_imagedatagenerator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator#__init__)

## Class ImageDataGenerator

Generate batches of tensor image data with real-time data augmentation.

### Aliases:

* Class tf.compat.v1.keras.preprocessing.image.ImageDataGenerator
* Class tf.compat.v2.keras.preprocessing.image.ImageDataGenerator
* Class tf.keras.preprocessing.image.ImageDataGenerator

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

### Used in the tutorials:

* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)

The data will be looped over (in batches).

#### Arguments:

* **featurewise\_center**: Boolean. Set input mean to 0 over the dataset, feature-wise.
* **samplewise\_center**: Boolean. Set each sample mean to 0.
* **featurewise\_std\_normalization**: Boolean. Divide inputs by std of the dataset, feature-wise.
* **samplewise\_std\_normalization**: Boolean. Divide each input by its std.
* **zca\_epsilon**: epsilon for ZCA whitening. Default is 1e-6.
* **zca\_whitening**: Boolean. Apply ZCA whitening.
* **rotation\_range**: Int. Degree range for random rotations.
* **width\_shift\_range**: Float, 1-D array-like or int
  + float: fraction of total width, if < 1, or pixels if >= 1.
  + 1-D array-like: random elements from the array.
  + int: integer number of pixels from interval (-width\_shift\_range, +width\_shift\_range)
  + With width\_shift\_range=2 possible values are integers [-1, 0, +1], same as with width\_shift\_range=[-1, 0, +1], while with width\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0).
* **height\_shift\_range**: Float, 1-D array-like or int
  + float: fraction of total height, if < 1, or pixels if >= 1.
  + 1-D array-like: random elements from the array.
  + int: integer number of pixels from interval (-height\_shift\_range, +height\_shift\_range)
  + With height\_shift\_range=2 possible values are integers [-1, 0, +1], same as with height\_shift\_range=[-1, 0, +1], while with height\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0).
* **brightness\_range**: Tuple or list of two floats. Range for picking a brightness shift value from.
* **shear\_range**: Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees)
* **zoom\_range**: Float or [lower, upper]. Range for random zoom. If a float, [lower, upper] = [1-zoom\_range, 1+zoom\_range].
* **channel\_shift\_range**: Float. Range for random channel shifts.
* **fill\_mode**: One of {"constant", "nearest", "reflect" or "wrap"}. Default is 'nearest'. Points outside the boundaries of the input are filled according to the given mode:
  + 'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)
  + 'nearest': aaaaaaaa|abcd|dddddddd
  + 'reflect': abcddcba|abcd|dcbaabcd
  + 'wrap': abcdabcd|abcd|abcdabcd
* **cval**: Float or Int. Value used for points outside the boundaries when fill\_mode = "constant".
* **horizontal\_flip**: Boolean. Randomly flip inputs horizontally.
* **vertical\_flip**: Boolean. Randomly flip inputs vertically.
* **rescale**: rescaling factor. Defaults to None. If None or 0, no rescaling is applied, otherwise we multiply the data by the value provided (after applying all other transformations).
* **preprocessing\_function**: function that will be implied on each input. The function will run after the image is resized and augmented. The function should take one argument: one image (Numpy tensor with rank 3), and should output a Numpy tensor with the same shape.
* **data\_format**: Image data format, either "channels\_first" or "channels\_last". "channels\_last" mode means that the images should have shape (samples, height, width, channels), "channels\_first" mode means that the images should have shape (samples, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **validation\_split**: Float. Fraction of images reserved for validation (strictly between 0 and 1).
* **dtype**: Dtype to use for the generated arrays.

#### Examples:

Example of using .flow(x, y):

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()  
y\_train = np\_utils.to\_categorical(y\_train, num\_classes)  
y\_test = np\_utils.to\_categorical(y\_test, num\_classes)  
datagen = ImageDataGenerator(  
    featurewise\_center=True,  
    featurewise\_std\_normalization=True,  
    rotation\_range=20,  
    width\_shift\_range=0.2,  
    height\_shift\_range=0.2,  
    horizontal\_flip=True)  
# compute quantities required for featurewise normalization  
# (std, mean, and principal components if ZCA whitening is applied)  
datagen.fit(x\_train)  
# fits the model on batches with real-time data augmentation:  
model.fit\_generator(datagen.flow(x\_train, y\_train, batch\_size=32),  
                    steps\_per\_epoch=len(x\_train) / 32, epochs=epochs)  
# here's a more "manual" example  
for e in range(epochs):  
    print('Epoch', e)  
    batches = 0  
    for x\_batch, y\_batch in datagen.flow(x\_train, y\_train, batch\_size=32):  
        model.fit(x\_batch, y\_batch)  
        batches += 1  
        if batches >= len(x\_train) / 32:  
            # we need to break the loop by hand because  
            # the generator loops indefinitely  
            break

Example of using .flow\_from\_directory(directory):

train\_datagen = ImageDataGenerator(  
        rescale=1./255,  
        shear\_range=0.2,  
        zoom\_range=0.2,  
        horizontal\_flip=True)  
test\_datagen = ImageDataGenerator(rescale=1./255)  
train\_generator = train\_datagen.flow\_from\_directory(  
        'data/train',  
        target\_size=(150, 150),  
        batch\_size=32,  
        class\_mode='binary')  
validation\_generator = test\_datagen.flow\_from\_directory(  
        'data/validation',  
        target\_size=(150, 150),  
        batch\_size=32,  
        class\_mode='binary')  
model.fit\_generator(  
        train\_generator,  
        steps\_per\_epoch=2000,  
        epochs=50,  
        validation\_data=validation\_generator,  
        validation\_steps=800)

Example of transforming images and masks together.

# we create two instances with the same **Arguments**  
data\_gen\_args = dict(featurewise\_center=True,  
                     featurewise\_std\_normalization=True,  
                     rotation\_range=90,  
                     width\_shift\_range=0.1,  
                     height\_shift\_range=0.1,  
                     zoom\_range=0.2)  
image\_datagen = ImageDataGenerator(\*\*data\_gen\_args)  
mask\_datagen = ImageDataGenerator(\*\*data\_gen\_args)  
# Provide the same seed and keyword **Arguments** to the fit and flow methods  
seed = 1  
image\_datagen.fit(images, augment=True, seed=seed)  
mask\_datagen.fit(masks, augment=True, seed=seed)  
image\_generator = image\_datagen.flow\_from\_directory(  
    'data/images',  
    class\_mode=None,  
    seed=seed)  
mask\_generator = mask\_datagen.flow\_from\_directory(  
    'data/masks',  
    class\_mode=None,  
    seed=seed)  
# combine generators into one which yields image and masks  
train\_generator = zip(image\_generator, mask\_generator)  
model.fit\_generator(  
    train\_generator,  
    steps\_per\_epoch=2000,  
    epochs=50)

## \_\_init\_\_

\_\_init\_\_(  
    featurewise\_center=False,  
    samplewise\_center=False,  
    featurewise\_std\_normalization=False,  
    samplewise\_std\_normalization=False,  
    zca\_whitening=False,  
    zca\_epsilon=1e-06,  
    rotation\_range=0,  
    width\_shift\_range=0.0,  
    height\_shift\_range=0.0,  
    brightness\_range=None,  
    shear\_range=0.0,  
    zoom\_range=0.0,  
    channel\_shift\_range=0.0,  
    fill\_mode='nearest',  
    cval=0.0,  
    horizontal\_flip=False,  
    vertical\_flip=False,  
    rescale=None,  
    preprocessing\_function=None,  
    data\_format=None,  
    validation\_split=0.0,  
    dtype=None  
)

## Methods

### apply\_transform

apply\_transform(  
    x,  
    transform\_parameters  
)

Applies a transformation to an image according to given parameters.

# Arguments

x: 3D tensor, single image.  
transform\_parameters: Dictionary with string - parameter pairs  
    describing the transformation.  
    Currently, the following parameters  
    from the dictionary are used:  
    - `'theta'`: Float. Rotation angle in degrees.  
    - `'tx'`: Float. Shift in the x direction.  
    - `'ty'`: Float. Shift in the y direction.  
    - `'shear'`: Float. Shear angle in degrees.  
    - `'zx'`: Float. Zoom in the x direction.  
    - `'zy'`: Float. Zoom in the y direction.  
    - `'flip\_horizontal'`: Boolean. Horizontal flip.  
    - `'flip\_vertical'`: Boolean. Vertical flip.  
    - `'channel\_shift\_intencity'`: Float. Channel shift intensity.  
    - `'brightness'`: Float. Brightness shift intensity.

# Returns

A transformed version of the input (same shape).

### fit

fit(  
    x,  
    augment=False,  
    rounds=1,  
    seed=None  
)

Fits the data generator to some sample data.

This computes the internal data stats related to the data-dependent transformations, based on an array of sample data.

Only required if featurewise\_center or featurewise\_std\_normalization or zca\_whitening are set to True.

# Arguments

x: Sample data. Should have rank 4.  
 In case of grayscale data,  
 the channels axis should have value 1, in case  
 of RGB data, it should have value 3, and in case  
 of RGBA data, it should have value 4.  
augment: Boolean (default: False).  
    Whether to fit on randomly augmented samples.  
rounds: Int (default: 1).  
    If using data augmentation (`augment=True`),  
    this is how many augmentation passes over the data to use.  
seed: Int (default: None). Random seed.

### flow

flow(  
    x,  
    y=None,  
    batch\_size=32,  
    shuffle=True,  
    sample\_weight=None,  
    seed=None,  
    save\_to\_dir=None,  
    save\_prefix='',  
    save\_format='png',  
    subset=None  
)

Takes data & label arrays, generates batches of augmented data.

# Arguments

x: Input data. Numpy array of rank 4 or a tuple.  
    If tuple, the first element  
    should contain the images and the second element  
    another numpy array or a list of numpy arrays  
    that gets passed to the output  
    without any modifications.  
    Can be used to feed the model miscellaneous data  
    along with the images.  
    In case of grayscale data, the channels axis of the image array  
    should have value 1, in case  
    of RGB data, it should have value 3, and in case  
    of RGBA data, it should have value 4.  
y: Labels.  
batch\_size: Int (default: 32).  
shuffle: Boolean (default: True).  
sample\_weight: Sample weights.  
seed: Int (default: None).  
save\_to\_dir: None or str (default: None).  
    This allows you to optionally specify a directory  
    to which to save the augmented pictures being generated  
    (useful for visualizing what you are doing).  
save\_prefix: Str (default: `''`).  
    Prefix to use for filenames of saved pictures  
    (only relevant if `save\_to\_dir` is set).  
save\_format: one of "png", "jpeg"  
    (only relevant if `save\_to\_dir` is set). Default: "png".  
subset: Subset of data (`"training"` or `"validation"`) if  
    `validation\_split` is set in `ImageDataGenerator`.

# Returns

An `Iterator` yielding tuples of `(x, y)`  
    where `x` is a numpy array of image data  
    (in the case of a single image input) or a list  
    of numpy arrays (in the case with  
    additional inputs) and `y` is a numpy array  
    of corresponding labels. If 'sample\_weight' is not None,  
    the yielded tuples are of the form `(x, y, sample\_weight)`.  
    If `y` is None, only the numpy array `x` is returned.

### flow\_from\_dataframe

flow\_from\_dataframe(  
    dataframe,  
    directory=None,  
    x\_col='filename',  
    y\_col='class',  
    weight\_col=None,  
    target\_size=(256, 256),  
    color\_mode='rgb',  
    classes=None,  
    class\_mode='categorical',  
    batch\_size=32,  
    shuffle=True,  
    seed=None,  
    save\_to\_dir=None,  
    save\_prefix='',  
    save\_format='png',  
    subset=None,  
    interpolation='nearest',  
    validate\_filenames=True,  
    \*\*kwargs  
)

Takes the dataframe and the path to a directory and generates batches of augmented/normalized data.

\*\*A simple tutorial can be found \*\*[here](http://bit.ly/keras_flow_from_dataframe).

# Arguments

dataframe: Pandas dataframe containing the filepaths relative to  
    `directory` (or absolute paths if `directory` is None) of the  
    images in a string column. It should include other column/s  
    depending on the `class\_mode`:  
    - if `class\_mode` is `"categorical"` (default value) it must  
        include the `y\_col` column with the class/es of each image.  
        Values in column can be string/list/tuple if a single class  
        or list/tuple if multiple classes.  
    - if `class\_mode` is `"binary"` or `"sparse"` it must include  
        the given `y\_col` column with class values as strings.  
    - if `class\_mode` is `"raw"` or `"multi\_output"` it should contain  
    the columns specified in `y\_col`.  
    - if `class\_mode` is `"input"` or `None` no extra column is needed.  
directory: string, path to the directory to read images from. If `None`,  
    data in `x\_col` column should be absolute paths.  
x\_col: string, column in `dataframe` that contains the filenames (or  
    absolute paths if `directory` is `None`).  
y\_col: string or list, column/s in `dataframe` that has the target data.  
weight\_col: string, column in `dataframe` that contains the sample  
    weights. Default: `None`.  
target\_size: tuple of integers `(height, width)`, default: `(256, 256)`.  
    The dimensions to which all images found will be resized.  
color\_mode: one of "grayscale", "rgb", "rgba". Default: "rgb".  
    Whether the images will be converted to have 1 or 3 color channels.  
classes: optional list of classes (e.g. `['dogs', 'cats']`).  
    Default: None. If not provided, the list of classes will be  
    automatically inferred from the `y\_col`,  
    which will map to the label indices, will be alphanumeric).  
    The dictionary containing the mapping from class names to class  
    indices can be obtained via the attribute `class\_indices`.  
class\_mode: one of "binary", "categorical", "input", "multi\_output",  
    "raw", sparse" or None. Default: "categorical".  
    Mode for yielding the targets:  
    - `"binary"`: 1D numpy array of binary labels,  
    - `"categorical"`: 2D numpy array of one-hot encoded labels.  
        Supports multi-label output.  
    - `"input"`: images identical to input images (mainly used to  
        work with autoencoders),  
    - `"multi\_output"`: list with the values of the different columns,  
    - `"raw"`: numpy array of values in `y\_col` column(s),  
    - `"sparse"`: 1D numpy array of integer labels,  
    - `None`, no targets are returned (the generator will only yield  
        batches of image data, which is useful to use in  
        `model.predict\_generator()`).  
batch\_size: size of the batches of data (default: 32).  
shuffle: whether to shuffle the data (default: True)  
seed: optional random seed for shuffling and transformations.  
save\_to\_dir: None or str (default: None).  
    This allows you to optionally specify a directory  
    to which to save the augmented pictures being generated  
    (useful for visualizing what you are doing).  
save\_prefix: str. Prefix to use for filenames of saved pictures  
    (only relevant if `save\_to\_dir` is set).  
save\_format: one of "png", "jpeg"  
    (only relevant if `save\_to\_dir` is set). Default: "png".  
follow\_links: whether to follow symlinks inside class subdirectories  
    (default: False).  
subset: Subset of data (`"training"` or `"validation"`) if  
    `validation\_split` is set in `ImageDataGenerator`.  
interpolation: Interpolation method used to resample the image if the  
    target size is different from that of the loaded image.  
    Supported methods are `"nearest"`, `"bilinear"`, and `"bicubic"`.  
    If PIL version 1.1.3 or newer is installed, `"lanczos"` is also  
    supported. If PIL version 3.4.0 or newer is installed, `"box"` and  
    `"hamming"` are also supported. By default, `"nearest"` is used.  
validate\_filenames: Boolean, whether to validate image filenames in  
    `x\_col`. If `True`, invalid images will be ignored. Disabling this  
    option can lead to speed-up in the execution of this function.  
    Default: `True`.

# Returns

A `DataFrameIterator` yielding tuples of `(x, y)`  
where `x` is a numpy array containing a batch  
of images with shape `(batch\_size, \*target\_size, channels)`  
and `y` is a numpy array of corresponding labels.

### flow\_from\_directory

flow\_from\_directory(  
    directory,  
    target\_size=(256, 256),  
    color\_mode='rgb',  
    classes=None,  
    class\_mode='categorical',  
    batch\_size=32,  
    shuffle=True,  
    seed=None,  
    save\_to\_dir=None,  
    save\_prefix='',  
    save\_format='png',  
    follow\_links=False,  
    subset=None,  
    interpolation='nearest'  
)

Takes the path to a directory & generates batches of augmented data.

# Arguments

directory: string, path to the target directory.  
    It should contain one subdirectory per class.  
    Any PNG, JPG, BMP, PPM or TIF images  
    inside each of the subdirectories directory tree  
    will be included in the generator.  
    See [this script](  
    https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d)  
    for more details.  
target\_size: Tuple of integers `(height, width)`,  
    default: `(256, 256)`.  
    The dimensions to which all images found will be resized.  
color\_mode: One of "grayscale", "rgb", "rgba". Default: "rgb".  
    Whether the images will be converted to  
    have 1, 3, or 4 channels.  
classes: Optional list of class subdirectories  
    (e.g. `['dogs', 'cats']`). Default: None.  
    If not provided, the list of classes will be automatically  
    inferred from the subdirectory names/structure  
    under `directory`, where each subdirectory will  
    be treated as a different class  
    (and the order of the classes, which will map to the label  
    indices, will be alphanumeric).  
    The dictionary containing the mapping from class names to class  
    indices can be obtained via the attribute `class\_indices`.  
class\_mode: One of "categorical", "binary", "sparse",  
    "input", or None. Default: "categorical".  
    Determines the type of label arrays that are returned:  
    - "categorical" will be 2D one-hot encoded labels,  
    - "binary" will be 1D binary labels,  
        "sparse" will be 1D integer labels,  
    - "input" will be images identical  
        to input images (mainly used to work with autoencoders).  
    - If None, no labels are returned  
      (the generator will only yield batches of image data,  
      which is useful to use with `model.predict\_generator()`).  
      Please note that in case of class\_mode None,  
      the data still needs to reside in a subdirectory  
      of `directory` for it to work correctly.  
batch\_size: Size of the batches of data (default: 32).  
shuffle: Whether to shuffle the data (default: True)  
    If set to False, sorts the data in alphanumeric order.  
seed: Optional random seed for shuffling and transformations.  
save\_to\_dir: None or str (default: None).  
    This allows you to optionally specify  
    a directory to which to save  
    the augmented pictures being generated  
    (useful for visualizing what you are doing).  
save\_prefix: Str. Prefix to use for filenames of saved pictures  
    (only relevant if `save\_to\_dir` is set).  
save\_format: One of "png", "jpeg"  
    (only relevant if `save\_to\_dir` is set). Default: "png".  
follow\_links: Whether to follow symlinks inside  
    class subdirectories (default: False).  
subset: Subset of data (`"training"` or `"validation"`) if  
    `validation\_split` is set in `ImageDataGenerator`.  
interpolation: Interpolation method used to  
    resample the image if the  
    target size is different from that of the loaded image.  
    Supported methods are `"nearest"`, `"bilinear"`,  
    and `"bicubic"`.  
    If PIL version 1.1.3 or newer is installed, `"lanczos"` is also  
    supported. If PIL version 3.4.0 or newer is installed,  
    `"box"` and `"hamming"` are also supported.  
    By default, `"nearest"` is used.

# Returns

A `DirectoryIterator` yielding tuples of `(x, y)`  
    where `x` is a numpy array containing a batch  
    of images with shape `(batch\_size, \*target\_size, channels)`  
    and `y` is a numpy array of corresponding labels.

### get\_random\_transform

get\_random\_transform(  
    img\_shape,  
    seed=None  
)

Generates random parameters for a transformation.

# Arguments

seed: Random seed.  
img\_shape: Tuple of integers.  
    Shape of the image that is transformed.

# Returns

A dictionary containing randomly chosen parameters describing the  
transformation.

### random\_transform

random\_transform(  
    x,  
    seed=None  
)

Applies a random transformation to an image.

# Arguments

x: 3D tensor, single image.  
seed: Random seed.

# Returns

A randomly transformed version of the input (same shape).

### standardize

standardize(x)

Applies the normalization configuration in-place to a batch of inputs.

x is changed in-place since the function is mainly used internally to standarize images and feed them to your network. If a copy of x would be created instead it would have a significant performance cost. If you want to apply this method without changing the input in-place you can call the method creating a copy before:

standarize(np.copy(x))

# Arguments

x: Batch of inputs to be normalized.

# Returns

The inputs, normalized.

# tf.keras.preprocessing.image.img\_to\_array

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/img_to_array#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/img_to_array#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/img_to_array#used_in_the_guide)

Converts a PIL Image instance to a Numpy array.

### Aliases:

* tf.compat.v1.keras.preprocessing.image.img\_to\_array
* tf.compat.v2.keras.preprocessing.image.img\_to\_array
* tf.keras.preprocessing.image.img\_to\_array

tf.keras.preprocessing.image.img\_to\_array(  
    img,  
    data\_format=None,  
    dtype=None  
)

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

#### Arguments:

* **img**: PIL Image instance.
* **data\_format**: Image data format, either "channels\_first" or "channels\_last".
* **dtype**: Dtype to use for the returned array.

#### Returns:

A 3D Numpy array.

#### Raises:

* **ValueError**: if invalid img or data\_format is passed.

tf.keras.preprocessing.image.Iterator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator#top_of_page)
* [Class Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator#class_iterator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator#methods)

Class Iterator

Inherits From: [Sequence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence)

Aliases:

* Class tf.compat.v1.keras.preprocessing.image.Iterator
* Class tf.compat.v2.keras.preprocessing.image.Iterator
* Class tf.keras.preprocessing.image.Iterator

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

\_\_init\_\_

\_\_init\_\_(  
    n,  
    batch\_size,  
    shuffle,  
    seed  
)

Methods

\_\_getitem\_\_

\_\_getitem\_\_(idx)

\_\_iter\_\_

\_\_iter\_\_()

\_\_len\_\_

\_\_len\_\_()

next

next()

For python 2.x.

**Returns**

The next batch.

on\_epoch\_end

on\_epoch\_end()

reset

reset()

Class Members

* white\_list\_formats

tf.keras.preprocessing.image.load\_img

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/load_img#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/load_img#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/load_img#used_in_the_guide)

Loads an image into PIL format.

Aliases:

* tf.compat.v1.keras.preprocessing.image.load\_img
* tf.compat.v2.keras.preprocessing.image.load\_img
* tf.keras.preprocessing.image.load\_img

tf.keras.preprocessing.image.load\_img(  
    path,  
    grayscale=False,  
    color\_mode='rgb',  
    target\_size=None,  
    interpolation='nearest'  
)

Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

**Arguments**

path: Path to image file.  
grayscale: DEPRECATED use `color\_mode="grayscale"`.  
color\_mode: One of "grayscale", "rgb", "rgba". Default: "rgb".  
    The desired image format.  
target\_size: Either `None` (default to original size)  
    or tuple of ints `(img\_height, img\_width)`.  
interpolation: Interpolation method used to resample the image if the  
    target size is different from that of the loaded image.  
    Supported methods are "nearest", "bilinear", and "bicubic".  
    If PIL version 1.1.3 or newer is installed, "lanczos" is also  
    supported. If PIL version 3.4.0 or newer is installed, "box" and  
    "hamming" are also supported. By default, "nearest" is used.

**Returns**

A PIL Image instance.

**Raises**

ImportError: if PIL is not available.  
ValueError: if interpolation method is not supported.

# tf.keras.preprocessing.image.NumpyArrayIterator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/NumpyArrayIterator#top_of_page)
* [Class NumpyArrayIterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/NumpyArrayIterator#class_numpyarrayiterator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/NumpyArrayIterator#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/NumpyArrayIterator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/NumpyArrayIterator#methods)

## Class NumpyArrayIterator

Iterator yielding data from a Numpy array.

Inherits From: [Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/Iterator)

### Aliases:

* Class tf.compat.v1.keras.preprocessing.image.NumpyArrayIterator
* Class tf.compat.v2.keras.preprocessing.image.NumpyArrayIterator
* Class tf.keras.preprocessing.image.NumpyArrayIterator

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

#### Arguments:

* **x**: Numpy array of input data or tuple. If tuple, the second elements is either another numpy array or a list of numpy arrays, each of which gets passed through as an output without any modifications.
* **y**: Numpy array of targets data.
* **image\_data\_generator**: Instance of ImageDataGenerator to use for random transformations and normalization.
* **batch\_size**: Integer, size of a batch.
* **shuffle**: Boolean, whether to shuffle the data between epochs.
* **sample\_weight**: Numpy array of sample weights.
* **seed**: Random seed for data shuffling.
* **data\_format**: String, one of channels\_first, channels\_last.
* **save\_to\_dir**: Optional directory where to save the pictures being yielded, in a viewable format. This is useful for visualizing the random transformations being applied, for debugging purposes.
* **save\_prefix**: String prefix to use for saving sample images (if save\_to\_dir is set).
* **save\_format**: Format to use for saving sample images (if save\_to\_dir is set).
* **subset**: Subset of data ("training" or "validation") if validation\_split is set in ImageDataGenerator.
* **dtype**: Dtype to use for the generated arrays.

## \_\_init\_\_

\_\_init\_\_(  
    x,  
    y,  
    image\_data\_generator,  
    batch\_size=32,  
    shuffle=False,  
    sample\_weight=None,  
    seed=None,  
    data\_format=None,  
    save\_to\_dir=None,  
    save\_prefix='',  
    save\_format='png',  
    subset=None,  
    dtype=None  
)

## Methods

### \_\_getitem\_\_

\_\_getitem\_\_(idx)

### \_\_iter\_\_

\_\_iter\_\_()

### \_\_len\_\_

\_\_len\_\_()

### next

next()

For python 2.x.

# Returns

The next batch.

### on\_epoch\_end

on\_epoch\_end()

### reset

reset()

## Class Members

* white\_list\_formats

tf.keras.preprocessing.image.random\_brightness

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_brightness#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_brightness#aliases)

Performs a random brightness shift.

Aliases:

* tf.compat.v1.keras.preprocessing.image.random\_brightness
* tf.compat.v2.keras.preprocessing.image.random\_brightness
* tf.keras.preprocessing.image.random\_brightness

tf.keras.preprocessing.image.random\_brightness(  
    x,  
    brightness\_range  
)

**Arguments**

x: Input tensor. Must be 3D.  
brightness\_range: Tuple of floats; brightness range.  
channel\_axis: Index of axis for channels in the input tensor.

**Returns**

Numpy image tensor.

**Raises**

ValueError if `brightness\_range` isn't a tuple.

tf.keras.preprocessing.image.random\_channel\_shift

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_channel_shift#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_channel_shift#aliases)

Performs a random channel shift.

Aliases:

* tf.compat.v1.keras.preprocessing.image.random\_channel\_shift
* tf.compat.v2.keras.preprocessing.image.random\_channel\_shift
* tf.keras.preprocessing.image.random\_channel\_shift

tf.keras.preprocessing.image.random\_channel\_shift(  
    x,  
    intensity\_range,  
    channel\_axis=0  
)

**Arguments**

x: Input tensor. Must be 3D.  
intensity\_range: Transformation intensity.  
channel\_axis: Index of axis for channels in the input tensor.

**Returns**

Numpy image tensor.

tf.keras.preprocessing.image.random\_rotation

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_rotation#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_rotation#aliases)

Performs a random rotation of a Numpy image tensor.

Aliases:

* tf.compat.v1.keras.preprocessing.image.random\_rotation
* tf.compat.v2.keras.preprocessing.image.random\_rotation
* tf.keras.preprocessing.image.random\_rotation

tf.keras.preprocessing.image.random\_rotation(  
    x,  
    rg,  
    row\_axis=1,  
    col\_axis=2,  
    channel\_axis=0,  
    fill\_mode='nearest',  
    cval=0.0,  
    interpolation\_order=1  
)

**Arguments**

x: Input tensor. Must be 3D.  
rg: Rotation range, in degrees.  
row\_axis: Index of axis for rows in the input tensor.  
col\_axis: Index of axis for columns in the input tensor.  
channel\_axis: Index of axis for channels in the input tensor.  
fill\_mode: Points outside the boundaries of the input  
    are filled according to the given mode  
    (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).  
cval: Value used for points outside the boundaries  
    of the input if `mode='constant'`.  
interpolation\_order: int, order of spline interpolation.  
    see `ndimage.interpolation.affine\_transform`

**Returns**

Rotated Numpy image tensor.

tf.keras.preprocessing.image.random\_shear

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_shear#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_shear#aliases)

Performs a random spatial shear of a Numpy image tensor.

Aliases:

* tf.compat.v1.keras.preprocessing.image.random\_shear
* tf.compat.v2.keras.preprocessing.image.random\_shear
* tf.keras.preprocessing.image.random\_shear

tf.keras.preprocessing.image.random\_shear(  
    x,  
    intensity,  
    row\_axis=1,  
    col\_axis=2,  
    channel\_axis=0,  
    fill\_mode='nearest',  
    cval=0.0,  
    interpolation\_order=1  
)

**Arguments**

x: Input tensor. Must be 3D.  
intensity: Transformation intensity in degrees.  
row\_axis: Index of axis for rows in the input tensor.  
col\_axis: Index of axis for columns in the input tensor.  
channel\_axis: Index of axis for channels in the input tensor.  
fill\_mode: Points outside the boundaries of the input  
    are filled according to the given mode  
    (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).  
cval: Value used for points outside the boundaries  
    of the input if `mode='constant'`.  
interpolation\_order: int, order of spline interpolation.  
    see `ndimage.interpolation.affine\_transform`

**Returns**

# Sheared Numpy image tensor. tf.keras.preprocessing.image.random\_shift

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_shift#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_shift#aliases)

Performs a random spatial shift of a Numpy image tensor.

Aliases:

* tf.compat.v1.keras.preprocessing.image.random\_shift
* tf.compat.v2.keras.preprocessing.image.random\_shift
* tf.keras.preprocessing.image.random\_shift

tf.keras.preprocessing.image.random\_shift(  
    x,  
    wrg,  
    hrg,  
    row\_axis=1,  
    col\_axis=2,  
    channel\_axis=0,  
    fill\_mode='nearest',  
    cval=0.0,  
    interpolation\_order=1  
)

**Arguments**

x: Input tensor. Must be 3D.  
wrg: Width shift range, as a float fraction of the width.  
hrg: Height shift range, as a float fraction of the height.  
row\_axis: Index of axis for rows in the input tensor.  
col\_axis: Index of axis for columns in the input tensor.  
channel\_axis: Index of axis for channels in the input tensor.  
fill\_mode: Points outside the boundaries of the input  
    are filled according to the given mode  
    (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).  
cval: Value used for points outside the boundaries  
    of the input if `mode='constant'`.  
interpolation\_order: int, order of spline interpolation.  
    see `ndimage.interpolation.affine\_transform`

**Returns**

Shifted Numpy image tensor.

tf.keras.preprocessing.image.random\_zoom

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_zoom#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/random_zoom#aliases)

Performs a random spatial zoom of a Numpy image tensor.

Aliases:

* tf.compat.v1.keras.preprocessing.image.random\_zoom
* tf.compat.v2.keras.preprocessing.image.random\_zoom
* tf.keras.preprocessing.image.random\_zoom

tf.keras.preprocessing.image.random\_zoom(  
    x,  
    zoom\_range,  
    row\_axis=1,  
    col\_axis=2,  
    channel\_axis=0,  
    fill\_mode='nearest',  
    cval=0.0,  
    interpolation\_order=1  
)

**Arguments**

x: Input tensor. Must be 3D.  
zoom\_range: Tuple of floats; zoom range for width and height.  
row\_axis: Index of axis for rows in the input tensor.  
col\_axis: Index of axis for columns in the input tensor.  
channel\_axis: Index of axis for channels in the input tensor.  
fill\_mode: Points outside the boundaries of the input  
    are filled according to the given mode  
    (one of `{'constant', 'nearest', 'reflect', 'wrap'}`).  
cval: Value used for points outside the boundaries  
    of the input if `mode='constant'`.  
interpolation\_order: int, order of spline interpolation.  
    see `ndimage.interpolation.affine\_transform`

Returns

Zoomed Numpy image tensor.

Raises

ValueError: if `zoom\_range` isn't a tuple.

# tf.keras.preprocessing.image.save\_img

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/save_img#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/image/save_img#aliases)

Saves an image stored as a Numpy array to a path or file object.

### Aliases:

* tf.compat.v1.keras.preprocessing.image.save\_img
* tf.compat.v2.keras.preprocessing.image.save\_img
* tf.keras.preprocessing.image.save\_img

tf.keras.preprocessing.image.save\_img(  
    path,  
    x,  
    data\_format=None,  
    file\_format=None,  
    scale=True,  
    \*\*kwargs  
)

Defined in [python/keras/preprocessing/image.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/image.py).

#### Arguments:

* **path**: Path or file object.
* **x**: Numpy array.
* **data\_format**: Image data format, either "channels\_first" or "channels\_last".
* **file\_format**: Optional file format override. If omitted, the format to use is determined from the filename extension. If a file object was used instead of a filename, this parameter should always be used.
* **scale**: Whether to rescale image values to be within [0, 255].
* **\*\*kwargs**: Additional keyword **Arguments** passed to PIL.Image.save().

Module: tf.keras.preprocessing.sequence

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence#functions)

Utilities for preprocessing sequence data.

Aliases:

* Module tf.compat.v2.keras.preprocessing.sequence
* Module tf.keras.preprocessing.sequence

Defined in [python/keras/api/\_v2/keras/preprocessing/sequence/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/preprocessing/sequence/__init__.py).

Classes

[class TimeseriesGenerator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator): Utility class for generating batches of temporal data.

Functions

[make\_sampling\_table(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/make_sampling_table): Generates a word rank-based probabilistic sampling table.

[pad\_sequences(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences): Pads sequences to the same length.

[skipgrams(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/skipgrams): Generates skipgram word pairs.

tf.keras.preprocessing.sequence.make\_sampling\_table

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/make_sampling_table#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/make_sampling_table#aliases)

Generates a word rank-based probabilistic sampling table.

Aliases:

* tf.compat.v1.keras.preprocessing.sequence.make\_sampling\_table
* tf.compat.v2.keras.preprocessing.sequence.make\_sampling\_table
* tf.keras.preprocessing.sequence.make\_sampling\_table

tf.keras.preprocessing.sequence.make\_sampling\_table(  
    size,  
    sampling\_factor=1e-05  
)

Used for generating the sampling\_table argument for skipgrams. sampling\_table[i] is the probability of sampling the word i-th most common word in a dataset (more common words should be sampled less frequently, for balance).

The sampling probabilities are generated according to the sampling distribution used in word2vec:

p(word) = (min(1, sqrt(word\_frequency / sampling\_factor) /  
    (word\_frequency / sampling\_factor)))

We assume that the word frequencies follow Zipf's law (s=1) to derive a numerical approximation of frequency(rank):

frequency(rank) ~ 1/(rank \* (log(rank) + gamma) + 1/2 - 1/(12\*rank)) where gamma is the Euler-Mascheroni constant.

**Arguments**

size: Int, number of possible words to sample.  
sampling\_factor: The sampling factor in the word2vec formula.

**Returns**

A 1D Numpy array of length `size` where the ith entry  
is the probability that a word of rank i should be sampled.

tf.keras.preprocessing.sequence.pad\_sequences

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences#used_in_the_tutorials)

Pads sequences to the same length.

Aliases:

* tf.compat.v1.keras.preprocessing.sequence.pad\_sequences
* tf.compat.v2.keras.preprocessing.sequence.pad\_sequences
* tf.keras.preprocessing.sequence.pad\_sequences

tf.keras.preprocessing.sequence.pad\_sequences(  
    sequences,  
    maxlen=None,  
    dtype='int32',  
    padding='pre',  
    truncating='pre',  
    value=0.0  
)

Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Text classification with movie reviews](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification)
* [Word embeddings](https://www.tensorflow.org/beta/tutorials/text/word_embeddings)

This function transforms a list of num\_samples sequences (lists of integers) into a 2D Numpy array of shape (num\_samples, num\_timesteps). num\_timesteps is either the maxlen argument if provided, or the length of the longest sequence otherwise.

Sequences that are shorter than num\_timesteps are padded with value at the end.

Sequences longer than num\_timesteps are truncated so that they fit the desired length. The position where padding or truncation happens is determined by the **Arguments** padding and truncating, respectively.

Pre-padding is the default.

**Arguments**

sequences: List of lists, where each element is a sequence.  
maxlen: Int, maximum length of all sequences.  
dtype: Type of the output sequences.  
    To pad sequences with variable length strings, you can use `object`.  
padding: String, 'pre' or 'post':  
    pad either before or after each sequence.  
truncating: String, 'pre' or 'post':  
    remove values from sequences larger than  
    `maxlen`, either at the beginning or at the end of the sequences.  
value: Float or String, padding value.

**Returns**

x: Numpy array with shape `(len(sequences), maxlen)`

**Raises**

ValueError: In case of invalid values for `truncating` or `padding`,  
    or in case of invalid shape for a `sequences` entry.

tf.keras.preprocessing.sequence.pad\_sequences

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences#used_in_the_tutorials)

Pads sequences to the same length.

Aliases:

* tf.compat.v1.keras.preprocessing.sequence.pad\_sequences
* tf.compat.v2.keras.preprocessing.sequence.pad\_sequences
* tf.keras.preprocessing.sequence.pad\_sequences

tf.keras.preprocessing.sequence.pad\_sequences(  
    sequences,  
    maxlen=None,  
    dtype='int32',  
    padding='pre',  
    truncating='pre',  
    value=0.0  
)

Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Text classification with movie reviews](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification)
* [Word embeddings](https://www.tensorflow.org/beta/tutorials/text/word_embeddings)

This function transforms a list of num\_samples sequences (lists of integers) into a 2D Numpy array of shape (num\_samples, num\_timesteps). num\_timesteps is either the maxlen argument if provided, or the length of the longest sequence otherwise.

Sequences that are shorter than num\_timesteps are padded with value at the end.

Sequences longer than num\_timesteps are truncated so that they fit the desired length. The position where padding or truncation happens is determined by the **Arguments** padding and truncating, respectively.

Pre-padding is the default.

**Arguments**

sequences: List of lists, where each element is a sequence.  
maxlen: Int, maximum length of all sequences.  
dtype: Type of the output sequences.  
    To pad sequences with variable length strings, you can use `object`.  
padding: String, 'pre' or 'post':  
    pad either before or after each sequence.  
truncating: String, 'pre' or 'post':  
    remove values from sequences larger than  
    `maxlen`, either at the beginning or at the end of the sequences.  
value: Float or String, padding value.

**Returns**

x: Numpy array with shape `(len(sequences), maxlen)`

**Raises**

ValueError: In case of invalid values for `truncating` or `padding`,  
    or in case of invalid shape for a `sequences` entry.

tf.keras.preprocessing.sequence.TimeseriesGenerator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator#top_of_page)
* [Class TimeseriesGenerator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator#class_timeseriesgenerator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator#methods)

Class TimeseriesGenerator

Utility class for generating batches of temporal data.

Inherits From: [Sequence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence)

Aliases:

* Class tf.compat.v1.keras.preprocessing.sequence.TimeseriesGenerator
* Class tf.compat.v2.keras.preprocessing.sequence.TimeseriesGenerator
* Class tf.keras.preprocessing.sequence.TimeseriesGenerator

Defined in [python/keras/preprocessing/sequence.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/preprocessing/sequence.py).

This class takes in a sequence of data-points gathered at equal intervals, along with time series parameters such as stride, length of history, etc., to produce batches for training/validation.

**Arguments**

data: Indexable generator (such as list or Numpy array)  
    containing consecutive data points (timesteps).  
    The data should be at 2D, and axis 0 is expected  
    to be the time dimension.  
targets: Targets corresponding to timesteps in `data`.  
    It should have same length as `data`.  
length: Length of the output sequences (in number of timesteps).  
sampling\_rate: Period between successive individual timesteps  
    within sequences. For rate `r`, timesteps  
    `data[i]`, `data[i-r]`, ... `data[i - length]`  
    are used for create a sample sequence.  
stride: Period between successive output sequences.  
    For stride `s`, consecutive output samples would  
    be centered around `data[i]`, `data[i+s]`, `data[i+2\*s]`, etc.  
start\_index: Data points earlier than `start\_index` will not be used  
    in the output sequences. This is useful to reserve part of the  
    data for test or validation.  
end\_index: Data points later than `end\_index` will not be used  
    in the output sequences. This is useful to reserve part of the  
    data for test or validation.  
shuffle: Whether to shuffle output samples,  
    or instead draw them in chronological order.  
reverse: Boolean: if `true`, timesteps in each output sample will be  
    in reverse chronological order.  
batch\_size: Number of timeseries samples in each batch  
    (except maybe the last one).

**Returns**

A [Sequence](/utils/#sequence) instance.

Examples

from keras.preprocessing.sequence import TimeseriesGenerator  
import numpy as np  
data = np.array([[i] for i in range(50)])  
targets = np.array([[i] for i in range(50)])  
data\_gen = TimeseriesGenerator(data, targets,  
                               length=10, sampling\_rate=2,  
                               batch\_size=2)  
assert len(data\_gen) == 20  
batch\_0 = data\_gen[0]  
x, y = batch\_0  
assert np.array\_equal(x,  
                      np.array([[[0], [2], [4], [6], [8]],  
                                [[1], [3], [5], [7], [9]]]))  
assert np.array\_equal(y,  
                      np.array([[10], [11]]))

\_\_init\_\_

\_\_init\_\_(  
    data,  
    targets,  
    length,  
    sampling\_rate=1,  
    stride=1,  
    start\_index=0,  
    end\_index=None,  
    shuffle=False,  
    reverse=False,  
    batch\_size=128  
)

Methods

\_\_getitem\_\_

\_\_getitem\_\_(index)

\_\_iter\_\_

\_\_iter\_\_()

Create a generator that iterate over the Sequence.

\_\_len\_\_

\_\_len\_\_()

get\_config

get\_config()

Returns the TimeseriesGenerator configuration as Python dictionary.

**Returns**

A Python dictionary with the TimeseriesGenerator configuration.

on\_epoch\_end

on\_epoch\_end()

Method called at the end of every epoch.

to\_json

to\_json(\*\*kwargs)

Returns a JSON string containing the timeseries generator configuration. To load a generator from a JSON string, usekeras.preprocessing.sequence.timeseries\_generator\_from\_json(json\_string).

**Arguments**

\*\*kwargs: Additional keyword **Arguments**  
    to be passed to `json.dumps()`.

**Returns**

A JSON string containing the tokenizer configuration.

Module: tf.keras.preprocessing.text

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text#functions)

Utilities for text input preprocessing.

Aliases:

* Module tf.compat.v2.keras.preprocessing.text
* Module tf.keras.preprocessing.text

Defined in [python/keras/api/\_v2/keras/preprocessing/text/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/preprocessing/text/__init__.py).

Classes

[class Tokenizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/Tokenizer): Text tokenization utility class.

Functions

[hashing\_trick(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/hashing_trick): Converts a text to a sequence of indexes in a fixed-size hashing space.

[one\_hot(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/one_hot): One-hot encodes a text into a list of word indexes of size n.

[text\_to\_word\_sequence(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/text_to_word_sequence): Converts a text to a sequence of words (or tokens).

tf.keras.preprocessing.text.hashing\_trick

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/hashing_trick#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/hashing_trick#aliases)

Converts a text to a sequence of indexes in a fixed-size hashing space.

Aliases:

* tf.compat.v1.keras.preprocessing.text.hashing\_trick
* tf.compat.v2.keras.preprocessing.text.hashing\_trick
* tf.keras.preprocessing.text.hashing\_trick

tf.keras.preprocessing.text.hashing\_trick(  
    text,  
    n,  
    hash\_function=None,  
    filters='!"#$%&()\*+,-./:;<=>?@[\\]^\_`{|}~\t\n',  
    lower=True,  
    split=' '  
)

**Arguments**

text: Input text (string).  
n: Dimension of the hashing space.  
hash\_function: defaults to python `hash` function, can be 'md5' or  
    any function that takes in input a string and returns a int.  
    Note that 'hash' is not a stable hashing function, so  
    it is not consistent across different runs, while 'md5'  
    is a stable hashing function.  
filters: list (or concatenation) of characters to filter out, such as  
    punctuation. Default: ``!"#$%&()\*+,-./:;<=>?@[\]^\_`{|}~\t\n``,  
    includes basic punctuation, tabs, and newlines.  
lower: boolean. Whether to set the text to lowercase.  
split: str. Separator for word splitting.

**Returns**

A list of integer word indices (unicity non-guaranteed).

0 is a reserved index that won't be assigned to any word.

Two or more words may be assigned to the same index, due to possible collisions by the hashing function. The [probability](https://en.wikipedia.org/wiki/Birthday_problem#Probability_table) of a collision is in relation to the dimension of the hashing space and the number of distinct objects.

tf.keras.preprocessing.text.one\_hot

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/one_hot#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/one_hot#aliases)

One-hot encodes a text into a list of word indexes of size n.

Aliases:

* tf.compat.v1.keras.preprocessing.text.one\_hot
* tf.compat.v2.keras.preprocessing.text.one\_hot
* tf.keras.preprocessing.text.one\_hot

tf.keras.preprocessing.text.one\_hot(  
    text,  
    n,  
    filters='!"#$%&()\*+,-./:;<=>?@[\\]^\_`{|}~\t\n',  
    lower=True,  
    split=' '  
)

This is a wrapper to the hashing\_trick function using hash as the hashing function; unicity of word to index mapping non-guaranteed.

**Arguments**

text: Input text (string).  
n: int. Size of vocabulary.  
filters: list (or concatenation) of characters to filter out, such as  
    punctuation. Default: ``!"#$%&()\*+,-./:;<=>?@[\]^\_`{|}~\t\n``,  
    includes basic punctuation, tabs, and newlines.  
lower: boolean. Whether to set the text to lowercase.  
split: str. Separator for word splitting.

**Returns**

List of integers in [1, n]. Each integer encodes a word  
(unicity non-guaranteed).

tf.keras.preprocessing.text.text\_to\_word\_sequence

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/text_to_word_sequence#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/text_to_word_sequence#aliases)

Converts a text to a sequence of words (or tokens).

Aliases:

* tf.compat.v1.keras.preprocessing.text.text\_to\_word\_sequence
* tf.compat.v2.keras.preprocessing.text.text\_to\_word\_sequence
* tf.keras.preprocessing.text.text\_to\_word\_sequence

tf.keras.preprocessing.text.text\_to\_word\_sequence(  
    text,  
    filters='!"#$%&()\*+,-./:;<=>?@[\\]^\_`{|}~\t\n',  
    lower=True,  
    split=' '  
)

**Arguments**

text: Input text (string).  
filters: list (or concatenation) of characters to filter out, such as  
    punctuation. Default: ``!"#$%&()\*+,-./:;<=>?@[\]^\_`{|}~\t\n``,  
    includes basic punctuation, tabs, and newlines.  
lower: boolean. Whether to convert the input to lowercase.  
split: str. Separator for word splitting.

**Returns**

A list of words (or tokens).

tf.keras.preprocessing.text.Tokenizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/Tokenizer#top_of_page)
* [Class Tokenizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/Tokenizer#class_tokenizer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/Tokenizer#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/Tokenizer#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/preprocessing/text/Tokenizer#__init__)

Class Tokenizer

Text tokenization utility class.

Aliases:

* Class tf.compat.v1.keras.preprocessing.text.Tokenizer
* Class tf.compat.v2.keras.preprocessing.text.Tokenizer
* Class tf.keras.preprocessing.text.Tokenizer

Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)

This class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf...

**Arguments**

num\_words: the maximum number of words to keep, based  
    on word frequency. Only the most common `num\_words-1` words will  
    be kept.  
filters: a string where each element is a character that will be  
    filtered from the texts. The default is all punctuation, plus  
    tabs and line breaks, minus the `'` character.  
lower: boolean. Whether to convert the texts to lowercase.  
split: str. Separator for word splitting.  
char\_level: if True, every character will be treated as a token.  
oov\_token: if given, it will be added to word\_index and used to  
    replace out-of-vocabulary words during text\_to\_sequence calls

By default, all punctuation is removed, turning the texts into space-separated sequences of words (words maybe include the ' character). These sequences are then split into lists of tokens. They will then be indexed or vectorized.

0 is a reserved index that won't be assigned to any word.

\_\_init\_\_

\_\_init\_\_(  
    num\_words=None,  
    filters='!"#$%&()\*+,-./:;<=>?@[\\]^\_`{|}~\t\n',  
    lower=True,  
    split=' ',  
    char\_level=False,  
    oov\_token=None,  
    document\_count=0,  
    \*\*kwargs  
)

Methods

fit\_on\_sequences

fit\_on\_sequences(sequences)

Updates internal vocabulary based on a list of sequences.

Required before using sequences\_to\_matrix (if fit\_on\_texts was never called).

**Arguments**

sequences: A list of sequence.  
    A "sequence" is a list of integer word indices.

fit\_on\_texts

fit\_on\_texts(texts)

Updates internal vocabulary based on a list of texts.

In the case where texts contains lists, we assume each entry of the lists to be a token.

Required before using texts\_to\_sequences or texts\_to\_matrix.

**Arguments**

texts: can be a list of strings,  
    a generator of strings (for memory-efficiency),  
    or a list of list of strings.

get\_config

get\_config()

Returns the tokenizer configuration as Python dictionary. The word count dictionaries used by the tokenizer get serialized into plain JSON, so that the configuration can be read by other projects.

**Returns**

A Python dictionary with the tokenizer configuration.

sequences\_to\_matrix

sequences\_to\_matrix(  
    sequences,  
    mode='binary'  
)

Converts a list of sequences into a Numpy matrix.

**Arguments**

sequences: list of sequences  
    (a sequence is a list of integer word indices).  
mode: one of "binary", "count", "tfidf", "freq"

**Returns**

A Numpy matrix.

**Raises**

ValueError: In case of invalid `mode` argument,  
    or if the Tokenizer requires to be fit to sample data.

sequences\_to\_texts

sequences\_to\_texts(sequences)

Transforms each sequence into a list of text.

Only top num\_words-1 most frequent words will be taken into account. Only words known by the tokenizer will be taken into account.

**Arguments**

sequences: A list of sequences (list of integers).

**Returns**

A list of texts (strings)

sequences\_to\_texts\_generator

sequences\_to\_texts\_generator(sequences)

Transforms each sequence in sequences to a list of texts(strings).

Each sequence has to a list of integers. In other words, sequences should be a list of sequences

Only top num\_words-1 most frequent words will be taken into account. Only words known by the tokenizer will be taken into account.

**Arguments**

sequences: A list of sequences.

**Yields**

Yields individual texts.

texts\_to\_matrix

texts\_to\_matrix(  
    texts,  
    mode='binary'  
)

Convert a list of texts to a Numpy matrix.

**Arguments**

texts: list of strings.  
mode: one of "binary", "count", "tfidf", "freq".

**Returns**

A Numpy matrix.

texts\_to\_sequences

texts\_to\_sequences(texts)

Transforms each text in texts to a sequence of integers.

Only top num\_words-1 most frequent words will be taken into account. Only words known by the tokenizer will be taken into account.

**Arguments**

texts: A list of texts (strings).

**Returns**

A list of sequences.

texts\_to\_sequences\_generator

texts\_to\_sequences\_generator(texts)

Transforms each text in texts to a sequence of integers.

Each item in texts can also be a list, in which case we assume each item of that list to be a token.

Only top num\_words-1 most frequent words will be taken into account. Only words known by the tokenizer will be taken into account.

**Arguments**

texts: A list of texts (strings).

**Yields**

Yields individual sequences.

to\_json

to\_json(\*\*kwargs)

Returns a JSON string containing the tokenizer configuration. To load a tokenizer from a JSON string, use keras.preprocessing.text.tokenizer\_from\_json(json\_string).

**Arguments**

\*\*kwargs: Additional keyword **Arguments**  
    to be passed to `json.dumps()`.

**Returns**

A JSON string containing the tokenizer configuration.

Module: tf.keras.regularizers

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers#functions)

Built-in regularizers.

Aliases:

* Module tf.compat.v2.keras.regularizers
* Module tf.keras.regularizers

Defined in [python/keras/api/\_v2/keras/regularizers/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/regularizers/__init__.py).

Classes

[class L1L2](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/L1L2): Regularizer for L1 and L2 regularization.

[class Regularizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer): Regularizer base class.

Functions

[deserialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/deserialize)

[get(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/get)

[l1(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1)

[l1\_l2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1_l2)

[l2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l2)

[serialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/serialize)

tf.keras.regularizers.deserialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/deserialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/deserialize#aliases)

Aliases:

* tf.compat.v1.keras.regularizers.deserialize
* tf.compat.v2.keras.regularizers.deserialize
* tf.keras.regularizers.deserialize

tf.keras.regularizers.deserialize(  
    config,  
    custom\_objects=None  
)

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

tf.keras.regularizers.get

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/get#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/get#aliases)

Aliases:

* tf.compat.v1.keras.regularizers.get
* tf.compat.v2.keras.regularizers.get
* tf.keras.regularizers.get

tf.keras.regularizers.get(identifier)

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

tf.keras.regularizers.l1

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1#used_in_the_guide)

Aliases:

* tf.compat.v1.keras.regularizers.l1
* tf.compat.v2.keras.regularizers.l1
* tf.keras.regularizers.l1

tf.keras.regularizers.l1(l=0.01)

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

Used in the guide:

* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)

# tf.keras.regularizers.L1L2

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/L1L2#top_of_page)
* [Class L1L2](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/L1L2#class_l1l2)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/L1L2#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/L1L2#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/L1L2#methods)

## Class L1L2

Regularizer for L1 and L2 regularization.

Inherits From: [Regularizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer)

### Aliases:

* Class tf.compat.v1.keras.regularizers.L1L2
* Class tf.compat.v2.keras.regularizers.L1L2
* Class tf.keras.regularizers.L1L2

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

#### Arguments:

* **l1**: Float; L1 regularization factor.
* **l2**: Float; L2 regularization factor.

## \_\_init\_\_

\_\_init\_\_(  
    l1=0.0,  
    l2=0.0  
)

## Methods

### \_\_call\_\_

\_\_call\_\_(x)

### from\_config

from\_config(  
    cls,  
    config  
)

### get\_config

get\_config()

tf.keras.regularizers.l1\_l2

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1_l2#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l1_l2#aliases)

Aliases:

* tf.compat.v1.keras.regularizers.l1\_l2
* tf.compat.v2.keras.regularizers.l1\_l2
* tf.keras.regularizers.l1\_l2

tf.keras.regularizers.l1\_l2(  
    l1=0.01,  
    l2=0.01  
)

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

tf.keras.regularizers.l2

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l2#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l2#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l2#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/l2#used_in_the_tutorials)

Aliases:

* tf.compat.v1.keras.regularizers.l2
* tf.compat.v2.keras.regularizers.l2
* tf.keras.regularizers.l2

tf.keras.regularizers.l2(l=0.01)

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Keras: A quick overview](https://www.tensorflow.org/beta/guide/keras/overview)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

Used in the tutorials:

* [Explore overfitting and underfitting](https://www.tensorflow.org/beta/tutorials/keras/overfit_and_underfit)

tf.keras.regularizers.Regularizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer#top_of_page)
* [Class Regularizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer#class_regularizer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer#methods)
  + [\_\_call\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer#__call__)
  + [from\_config](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/Regularizer#from_config)

Class Regularizer

Regularizer base class.

Aliases:

* Class tf.compat.v1.keras.regularizers.Regularizer
* Class tf.compat.v2.keras.regularizers.Regularizer
* Class tf.keras.regularizers.Regularizer

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

Methods

\_\_call\_\_

\_\_call\_\_(x)

from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

tf.keras.regularizers.serialize

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/serialize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/regularizers/serialize#aliases)

Aliases:

* tf.compat.v1.keras.regularizers.serialize
* tf.compat.v2.keras.regularizers.serialize
* tf.keras.regularizers.serialize

tf.keras.regularizers.serialize(regularizer)

Defined in [python/keras/regularizers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/regularizers.py).

Module: tf.keras.utils

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils#aliases)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils#functions)

Keras utilities.

Aliases:

* Module tf.compat.v2.keras.utils
* Module tf.keras.utils

Defined in [python/keras/api/\_v2/keras/utils/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/utils/__init__.py).

Classes

[class CustomObjectScope](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope): Provides a scope that changes to \_GLOBAL\_CUSTOM\_OBJECTS cannot escape.

[class GeneratorEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/GeneratorEnqueuer): Builds a queue out of a data generator.

[class HDF5Matrix](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/HDF5Matrix): Representation of HDF5 dataset to be used instead of a Numpy array.

[class OrderedEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/OrderedEnqueuer): Builds a Enqueuer from a Sequence.

[class Progbar](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar): Displays a progress bar.

[class Sequence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence): Base object for fitting to a sequence of data, such as a dataset.

[class SequenceEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer): Base class to enqueue inputs.

Functions

[convert\_all\_kernels\_in\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/convert_all_kernels_in_model): Converts all convolution kernels in a model from Theano to TensorFlow.

[custom\_object\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/custom_object_scope): Provides a scope that changes to \_GLOBAL\_CUSTOM\_OBJECTS cannot escape.

[deserialize\_keras\_object(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/deserialize_keras_object)

[get\_custom\_objects(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_custom_objects): Retrieves a live reference to the global dictionary of custom objects.

[get\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_file): Downloads a file from a URL if it not already in the cache.

[get\_source\_inputs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_source_inputs): Returns the list of input tensors necessary to compute tensor.

[model\_to\_dot(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/model_to_dot): Convert a Keras model to dot format.

[multi\_gpu\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/multi_gpu_model): Replicates a model on different GPUs.

[normalize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/normalize): Normalizes a Numpy array.

[plot\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/plot_model): Converts a Keras model to dot format and save to a file.

[serialize\_keras\_object(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/serialize_keras_object)

[to\_categorical(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/to_categorical): Converts a class vector (integers) to binary class matrix.

# tf.keras.utils.convert\_all\_kernels\_in\_model

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/convert_all_kernels_in_model#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/convert_all_kernels_in_model#aliases)

Converts all convolution kernels in a model from Theano to TensorFlow.

### Aliases:

* tf.compat.v1.keras.utils.convert\_all\_kernels\_in\_model
* tf.compat.v2.keras.utils.convert\_all\_kernels\_in\_model
* tf.keras.utils.convert\_all\_kernels\_in\_model

tf.keras.utils.convert\_all\_kernels\_in\_model(model)

Defined in [python/keras/utils/layer\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/layer_utils.py).

Also works from TensorFlow to Theano.

#### Arguments:

* **model**: target model for the conversion.

# tf.keras.utils.CustomObjectScope

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#top_of_page)
* [Class CustomObjectScope](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#class_customobjectscope)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#methods)
  + [\_\_enter\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#__enter__)
  + [\_\_exit\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/CustomObjectScope#__exit__)

## Class CustomObjectScope

Provides a scope that changes to \_GLOBAL\_CUSTOM\_OBJECTS cannot escape.

### Aliases:

* Class tf.compat.v1.keras.utils.CustomObjectScope
* Class tf.compat.v2.keras.utils.CustomObjectScope
* Class tf.keras.utils.CustomObjectScope

Defined in [python/keras/utils/generic\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/generic_utils.py).

Code within a with statement will be able to access custom objects by name. Changes to global custom objects persist within the enclosing with statement. At end of the with statement, global custom objects are reverted to state at beginning of the with statement.

#### Example:

Consider a custom object MyObject (e.g. a class):

    with CustomObjectScope({'MyObject':MyObject}):  
        layer = Dense(..., kernel\_regularizer='MyObject')  
        # save, load, etc. will recognize custom object by name

## \_\_init\_\_

\_\_init\_\_(\*args)

## Methods

### \_\_enter\_\_

\_\_enter\_\_()

### \_\_exit\_\_

\_\_exit\_\_(  
    \*args,  
    \*\*kwargs  
)

# tf.keras.utils.custom\_object\_scope

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/custom_object_scope#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/custom_object_scope#aliases)

Provides a scope that changes to \_GLOBAL\_CUSTOM\_OBJECTS cannot escape.

### Aliases:

* tf.compat.v1.keras.utils.custom\_object\_scope
* tf.compat.v2.keras.utils.custom\_object\_scope
* tf.keras.utils.custom\_object\_scope

tf.keras.utils.custom\_object\_scope(\*args)

Defined in [python/keras/utils/generic\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/generic_utils.py).

Convenience wrapper for CustomObjectScope. Code within a with statement will be able to access custom objects by name. Changes to global custom objects persist within the enclosing withstatement. At end of the with statement, global custom objects are reverted to state at beginning of the with statement.

#### Example:

Consider a custom object MyObject

    with custom\_object\_scope({'MyObject':MyObject}):  
        layer = Dense(..., kernel\_regularizer='MyObject')  
        # save, load, etc. will recognize custom object by name

#### Arguments:

* **\*args**: Variable length list of dictionaries of name, class pairs to add to custom objects.

#### Returns:

Object of type CustomObjectScope.

tf.keras.utils.deserialize\_keras\_object

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/deserialize_keras_object#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/deserialize_keras_object#aliases)

Aliases:

* tf.compat.v1.keras.utils.deserialize\_keras\_object
* tf.compat.v2.keras.utils.deserialize\_keras\_object
* tf.keras.utils.deserialize\_keras\_object

tf.keras.utils.deserialize\_keras\_object(  
    identifier,  
    module\_objects=None,  
    custom\_objects=None,  
    printable\_module\_name='object'  
)

Defined in [python/keras/utils/generic\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/generic_utils.py).

# tf.keras.utils.GeneratorEnqueuer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/GeneratorEnqueuer#top_of_page)
* [Class GeneratorEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/GeneratorEnqueuer#class_generatorenqueuer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/GeneratorEnqueuer#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/GeneratorEnqueuer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/GeneratorEnqueuer#methods)

## Class GeneratorEnqueuer

Builds a queue out of a data generator.

Inherits From: [SequenceEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer)

### Aliases:

* Class tf.compat.v1.keras.utils.GeneratorEnqueuer
* Class tf.compat.v2.keras.utils.GeneratorEnqueuer
* Class tf.keras.utils.GeneratorEnqueuer

Defined in [python/keras/utils/data\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/data_utils.py).

The provided generator can be finite in which case the class will throw a StopIteration exception.

Used in fit\_generator, evaluate\_generator, predict\_generator.

#### Arguments:

* **generator**: a generator function which yields data
* **use\_multiprocessing**: use multiprocessing if True, otherwise threading
* **wait\_time**: time to sleep in-between calls to put()
* **random\_seed**: Initial seed for workers, will be incremented by one for each worker.

## \_\_init\_\_

\_\_init\_\_(  
    sequence,  
    use\_multiprocessing=False,  
    random\_seed=None  
)

## Methods

### get

get()

Creates a generator to extract data from the queue.

Skip the data if it is None.

#### Yields:

The next element in the queue, i.e. a tuple (inputs, targets) or (inputs, targets, sample\_weights).

### is\_running

is\_running()

### start

start(  
    workers=1,  
    max\_queue\_size=10  
)

Starts the handler's workers.

#### Arguments:

* **workers**: Number of workers.
* **max\_queue\_size**: queue size (when full, workers could block on put())

### stop

stop(timeout=None)

Stops running threads and wait for them to exit, if necessary.

Should be called by the same thread which called start().

#### Arguments:

* **timeout**: maximum time to wait on thread.join()

# tf.keras.utils.get\_custom\_objects

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_custom_objects#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_custom_objects#aliases)

Retrieves a live reference to the global dictionary of custom objects.

### Aliases:

* tf.compat.v1.keras.utils.get\_custom\_objects
* tf.compat.v2.keras.utils.get\_custom\_objects
* tf.keras.utils.get\_custom\_objects

tf.keras.utils.get\_custom\_objects()

Defined in [python/keras/utils/generic\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/generic_utils.py).

Updating and clearing custom objects using custom\_object\_scope is preferred, but get\_custom\_objects can be used to directly access \_GLOBAL\_CUSTOM\_OBJECTS.

#### Example:

    get\_custom\_objects().clear()  
    get\_custom\_objects()['MyObject'] = MyObject

#### Returns:

Global dictionary of names to classes (\_GLOBAL\_CUSTOM\_OBJECTS).

# tf.keras.utils.get\_file

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_file#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_file#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_file#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_file#used_in_the_tutorials)

Downloads a file from a URL if it not already in the cache.

### Aliases:

* tf.compat.v1.keras.utils.get\_file
* tf.compat.v2.keras.utils.get\_file
* tf.keras.utils.get\_file

tf.keras.utils.get\_file(  
    fname,  
    origin,  
    untar=False,  
    md5\_hash=None,  
    file\_hash=None,  
    cache\_subdir='datasets',  
    hash\_algorithm='auto',  
    extract=False,  
    archive\_format='auto',  
    cache\_dir=None  
)

Defined in [python/keras/utils/data\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/data_utils.py).

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load CSV with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/csv)
* [Load NumPy Data with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/numpy)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Load text with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/text)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Premade Estimators](https://www.tensorflow.org/beta/tutorials/estimators/premade_estimators)
* [Regression: Predict fuel efficiency](https://www.tensorflow.org/beta/tutorials/keras/basic_regression)
* [TensorFlow Hub with Keras](https://www.tensorflow.org/beta/tutorials/images/hub_with_keras)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

By default the file at the url origin is downloaded to the cache\_dir ~/.keras, placed in the cache\_subdir datasets, and given the filename fname. The final location of a file example.txtwould therefore be ~/.keras/datasets/example.txt.

Files in tar, tar.gz, tar.bz, and zip formats can also be extracted. Passing a hash will verify the file after download. The command line programs shasum and sha256sum can compute the hash.

#### Arguments:

**fname**: Name of the file. If an absolute path /path/to/file.txt is specified the file will be saved at that location.

**origin**: Original URL of the file.

**untar**: Deprecated in favor of 'extract'. boolean, whether the file should be decompressed

**md5\_hash**: Deprecated in favor of 'file\_hash'. md5 hash of the file for verification

**file\_hash**: The expected hash string of the file after download. The sha256 and md5 hash algorithms are both supported.

**cache\_subdir**: Subdirectory under the Keras cache dir where the file is saved. If an absolute path /path/to/folder is specified the file will be saved at that location.

**hash\_algorithm**: Select the hash algorithm to verify the file. options are 'md5', 'sha256', and 'auto'. The default 'auto' detects the hash algorithm in use.

**extract**: True tries extracting the file as an Archive, like tar or zip.

**archive\_format**: Archive format to try for extracting the file. Options are 'auto', 'tar', 'zip', and None. 'tar' includes tar, tar.gz, and tar.bz files. The default 'auto' is ['tar', 'zip']. None or an empty list will return no matches found.

**cache\_dir**: Location to store cached files, when None it defaults to the [Keras Directory](https://www.tensorflow.org/faq/#where-is-the-keras-configuration-filed-stored).

#### Returns:

Path to the downloaded file

# tf.keras.utils.get\_source\_inputs

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_source_inputs#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/get_source_inputs#aliases)

Returns the list of input tensors necessary to compute tensor.

### Aliases:

tf.compat.v1.keras.utils.get\_source\_inputs

tf.compat.v2.keras.utils.get\_source\_inputs

tf.keras.utils.get\_source\_inputs

tf.keras.utils.get\_source\_inputs(  
    tensor,  
    layer=None,  
    node\_index=None  
)

Defined in [python/keras/utils/layer\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/layer_utils.py).

Output will always be a list of tensors (potentially with 1 element).

#### Arguments:

**tensor**: The tensor to start from.

**layer**: Origin layer of the tensor. Will be determined via tensor.\_keras\_history if not provided.

**node\_index**: Origin node index of the tensor.

#### Returns:

List of input tensors.

# tf.keras.utils.HDF5Matrix

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/HDF5Matrix#top_of_page)

[Class HDF5Matrix](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/HDF5Matrix#class_hdf5matrix)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/HDF5Matrix#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/HDF5Matrix#__init__)

[Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/HDF5Matrix#properties)

## Class HDF5Matrix

Representation of HDF5 dataset to be used instead of a Numpy array.

### Aliases:

Class tf.compat.v1.keras.utils.HDF5Matrix

Class tf.compat.v2.keras.utils.HDF5Matrix

Class tf.keras.utils.HDF5Matrix

Defined in [python/keras/utils/io\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/io_utils.py).

#### Example:

    x\_data = HDF5Matrix('input/file.hdf5', 'data')  
    model.predict(x\_data)

Providing start and end allows use of a slice of the dataset.

Optionally, a normalizer function (or lambda) can be given. This will be called on every slice of data retrieved.

#### Arguments:

**datapath**: string, path to a HDF5 file

**dataset**: string, name of the HDF5 dataset in the file specified in datapath

**start**: int, start of desired slice of the specified dataset

**end**: int, end of desired slice of the specified dataset

**normalizer**: function to be called on data when retrieved

#### Returns:

An array-like HDF5 dataset.

## \_\_init\_\_

\_\_init\_\_(  
    datapath,  
    dataset,  
    start=0,  
    end=None,  
    normalizer=None  
)

## Properties

### dtype

Gets the datatype of the dataset.

#### Returns:

A numpy dtype string.

### ndim

Gets the number of dimensions (rank) of the dataset.

#### Returns:

An integer denoting the number of dimensions (rank) of the dataset.

### shape

Gets a numpy-style shape tuple giving the dataset dimensions.

#### Returns:

A numpy-style shape tuple.

### size

Gets the total dataset size (number of elements).

#### Returns:

An integer denoting the number of elements in the dataset.

## Methods

### \_\_getitem\_\_

\_\_getitem\_\_(key)

### \_\_len\_\_

\_\_len\_\_()

## Class Members

refs

# tf.keras.utils.model\_to\_dot

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/model_to_dot#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/model_to_dot#aliases)

Convert a Keras model to dot format.

### Aliases:

tf.compat.v1.keras.utils.model\_to\_dot

tf.compat.v2.keras.utils.model\_to\_dot

tf.keras.utils.model\_to\_dot

tf.keras.utils.model\_to\_dot(  
    model,  
    show\_shapes=False,  
    show\_layer\_names=True,  
    rankdir='TB',  
    expand\_nested=False,  
    dpi=96,  
    subgraph=False  
)

Defined in [python/keras/utils/vis\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/vis_utils.py).

#### Arguments:

**model**: A Keras model instance.

**show\_shapes**: whether to display shape information.

**show\_layer\_names**: whether to display layer names.

**rankdir**: rankdir argument passed to PyDot, a string specifying the format of the plot: 'TB' creates a vertical plot; 'LR' creates a horizontal plot.

**expand\_nested**: whether to expand nested models into clusters.

**dpi**: Dots per inch.

**subgraph**: whether to return a pydot.Cluster instance.

#### Returns:

A pydot.Dot instance representing the Keras model or a pydot.Cluster instance representing nested model if subgraph=True.

#### Raises:

**ImportError**: if graphviz or pydot are not available.

# tf.keras.utils.multi\_gpu\_model

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/multi_gpu_model#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/multi_gpu_model#aliases)

Replicates a model on different GPUs.

### Aliases:

tf.compat.v1.keras.utils.multi\_gpu\_model

tf.compat.v2.keras.utils.multi\_gpu\_model

tf.keras.utils.multi\_gpu\_model

tf.keras.utils.multi\_gpu\_model(  
    model,  
    gpus,  
    cpu\_merge=True,  
    cpu\_relocation=False  
)

Defined in [python/keras/utils/multi\_gpu\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/multi_gpu_utils.py).

Specifically, this function implements single-machine multi-GPU data parallelism. It works in the following way:

Divide the model's input(s) into multiple sub-batches.

Apply a model copy on each sub-batch. Every model copy is executed on a dedicated GPU.

Concatenate the results (on CPU) into one big batch.

E.g. if your batch\_size is 64 and you use gpus=2, then we will divide the input into 2 sub-batches of 32 samples, process each sub-batch on one GPU, then return the full batch of 64 processed samples.

This induces quasi-linear speedup on up to 8 GPUs.

This function is only available with the TensorFlow backend for the time being.

#### Arguments:

**model**: A Keras model instance. To avoid OOM errors, this model could have been built on CPU, for instance (see usage example below).

**gpus**: Integer >= 2, number of on GPUs on which to create model replicas.

**cpu\_merge**: A boolean value to identify whether to force merging model weights under the scope of the CPU or not.

**cpu\_relocation**: A boolean value to identify whether to create the model's weights under the scope of the CPU. If the model is not defined under any preceding device scope, you can still rescue it by activating this option.

#### Returns:

A Keras Model instance which can be used just like the initial model argument, but which distributes its workload on multiple GPUs.

Example 1: Training models with weights merge on CPU

    import tensorflow as tf  
    from keras.applications import Xception  
    from keras.utils import multi\_gpu\_model  
    import numpy as np  
  
    num\_samples = 1000  
    height = 224  
    width = 224  
    num\_classes = 1000  
  
    # Instantiate the base model (or "template" model).  
    # We recommend doing this with under a CPU device scope,  
    # so that the model's weights are hosted on CPU memory.  
    # Otherwise they may end up hosted on a GPU, which would  
    # complicate weight sharing.  
    with tf.device('/cpu:0'):  
        model = Xception(weights=None,  
                         input\_shape=(height, width, 3),  
                         classes=num\_classes)  
  
    # Replicates the model on 8 GPUs.  
    # This assumes that your machine has 8 available GPUs.  
    parallel\_model = multi\_gpu\_model(model, gpus=8)  
    parallel\_model.compile(loss='categorical\_crossentropy',  
                           optimizer='rmsprop')  
  
    # Generate dummy data.  
    x = np.random.random((num\_samples, height, width, 3))  
    y = np.random.random((num\_samples, num\_classes))  
  
    # This `fit` call will be distributed on 8 GPUs.  
    # Since the batch size is 256, each GPU will process 32 samples.  
    parallel\_model.fit(x, y, epochs=20, batch\_size=256)  
  
    # Save model via the template model (which shares the same weights):  
    model.save('my\_model.h5')

Example 2: Training models with weights merge on CPU using cpu\_relocation

     ..  
     # Not needed to change the device scope for model definition:  
     model = Xception(weights=None, ..)  
  
     try:  
         model = multi\_gpu\_model(model, cpu\_relocation=True)  
         print("Training using multiple GPUs..")  
     except:  
         print("Training using single GPU or CPU..")  
  
     model.compile(..)  
     ..

Example 3: Training models with weights merge on GPU (recommended for NV-link)

     ..  
     # Not needed to change the device scope for model definition:  
     model = Xception(weights=None, ..)  
  
     try:  
         model = multi\_gpu\_model(model, cpu\_merge=False)  
         print("Training using multiple GPUs..")  
     except:  
         print("Training using single GPU or CPU..")  
     model.compile(..)  
     ..

#### Raises:

**ValueError**: if the gpus argument does not match available devices.

# tf.keras.utils.normalize

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/normalize#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/normalize#aliases)

Normalizes a Numpy array.

### Aliases:

tf.compat.v1.keras.utils.normalize

tf.compat.v2.keras.utils.normalize

tf.keras.utils.normalize

tf.keras.utils.normalize(  
    x,  
    axis=-1,  
    order=2  
)

Defined in [python/keras/utils/np\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/np_utils.py).

#### Arguments:

**x**: Numpy array to normalize.

**axis**: axis along which to normalize.

**order**: Normalization order (e.g. 2 for L2 norm).

#### Returns:

A normalized copy of the array.

# tf.keras.utils.OrderedEnqueuer

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/OrderedEnqueuer#top_of_page)

[Class OrderedEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/OrderedEnqueuer#class_orderedenqueuer)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/OrderedEnqueuer#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/OrderedEnqueuer#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/OrderedEnqueuer#methods)

## Class OrderedEnqueuer

Builds a Enqueuer from a Sequence.

Inherits From: [SequenceEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer)

### Aliases:

Class tf.compat.v1.keras.utils.OrderedEnqueuer

Class tf.compat.v2.keras.utils.OrderedEnqueuer

Class tf.keras.utils.OrderedEnqueuer

Defined in [python/keras/utils/data\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/data_utils.py).

Used in fit\_generator, evaluate\_generator, predict\_generator.

#### Arguments:

**sequence**: A tf.keras.utils.data\_utils.Sequence object.

**use\_multiprocessing**: use multiprocessing if True, otherwise threading

**shuffle**: whether to shuffle the data at the beginning of each epoch

## \_\_init\_\_

\_\_init\_\_(  
    sequence,  
    use\_multiprocessing=False,  
    shuffle=False  
)

## Methods

### get

get()

Creates a generator to extract data from the queue.

Skip the data if it is None.

#### Yields:

The next element in the queue, i.e. a tuple (inputs, targets) or (inputs, targets, sample\_weights).

### is\_running

is\_running()

### start

start(  
    workers=1,  
    max\_queue\_size=10  
)

Starts the handler's workers.

#### Arguments:

**workers**: Number of workers.

**max\_queue\_size**: queue size (when full, workers could block on put())

### stop

stop(timeout=None)

Stops running threads and wait for them to exit, if necessary.

Should be called by the same thread which called start().

#### Arguments:

**timeout**: maximum time to wait on thread.join()

# tf.keras.utils.plot\_model

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/plot_model#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/plot_model#aliases)

[Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/plot_model#used_in_the_guide)

Converts a Keras model to dot format and save to a file.

### Aliases:

tf.compat.v1.keras.utils.plot\_model

tf.compat.v2.keras.utils.plot\_model

tf.keras.utils.plot\_model

tf.keras.utils.plot\_model(  
    model,  
    to\_file='model.png',  
    show\_shapes=False,  
    show\_layer\_names=True,  
    rankdir='TB',  
    expand\_nested=False,  
    dpi=96  
)

Defined in [python/keras/utils/vis\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/vis_utils.py).

### Used in the guide:

[The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

[Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

#### Arguments:

**model**: A Keras model instance

**to\_file**: File name of the plot image.

**show\_shapes**: whether to display shape information.

**show\_layer\_names**: whether to display layer names.

**rankdir**: rankdir argument passed to PyDot, a string specifying the format of the plot: 'TB' creates a vertical plot; 'LR' creates a horizontal plot.

**expand\_nested**: Whether to expand nested models into clusters.

**dpi**: Dots per inch.

#### Returns:

A Jupyter notebook Image object if Jupyter is installed. This enables in-line display of the model plots in notebooks.

# tf.keras.utils.Progbar

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#top_of_page)

[Class Progbar](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#class_progbar)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#methods)

[add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#add)

[update](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Progbar#update)

## Class Progbar

Displays a progress bar.

### Aliases:

Class tf.compat.v1.keras.utils.Progbar

Class tf.compat.v2.keras.utils.Progbar

Class tf.keras.utils.Progbar

Defined in [python/keras/utils/generic\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/generic_utils.py).

#### Arguments:

**target**: Total number of steps expected, None if unknown.

**width**: Progress bar width on screen.

**verbose**: Verbosity mode, 0 (silent), 1 (verbose), 2 (semi-verbose)

**stateful\_metrics**: Iterable of string names of metrics that should not be averaged over time. Metrics in this list will be displayed as-is. All others will be averaged by the progbar before display.

**interval**: Minimum visual progress update interval (in seconds).

**unit\_name**: Display name for step counts (usually "step" or "sample").

## \_\_init\_\_

\_\_init\_\_(  
    target,  
    width=30,  
    verbose=1,  
    interval=0.05,  
    stateful\_metrics=None,  
    unit\_name='step'  
)

## Methods

### add

add(  
    n,  
    values=None  
)

### update

update(  
    current,  
    values=None  
)

Updates the progress bar.

#### Arguments:

**current**: Index of current step.

**values**: List of tuples: (name, value\_for\_last\_step). If name is in stateful\_metrics,value\_for\_last\_step will be displayed as-is. Else, an average of the metric over time will be displayed.

# tf.keras.utils.Sequence

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence#top_of_page)

[Class Sequence](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence#class_sequence)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence#aliases)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence#methods)

[\_\_getitem\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/Sequence#__getitem__)

## Class Sequence

Base object for fitting to a sequence of data, such as a dataset.

### Aliases:

Class tf.compat.v1.keras.utils.Sequence

Class tf.compat.v2.keras.utils.Sequence

Class tf.keras.utils.Sequence

Defined in [python/keras/utils/data\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/data_utils.py).

Every Sequence must implement the \_\_getitem\_\_ and the \_\_len\_\_ methods. If you want to modify your dataset between epochs you may implement on\_epoch\_end. The method \_\_getitem\_\_ should return a complete batch.

#### Notes:

Sequence are a safer way to do multiprocessing. This structure guarantees that the network will only train once on each sample per epoch which is not the case with generators.

#### Examples:

    from skimage.io import imread  
    from skimage.transform import resize  
    import numpy as np  
    import math  
  
    # Here, `x\_set` is list of path to the images  
    # and `y\_set` are the associated classes.  
  
    class CIFAR10Sequence(Sequence):  
  
        def \_\_init\_\_(self, x\_set, y\_set, batch\_size):  
            self.x, self.y = x\_set, y\_set  
            self.batch\_size = batch\_size  
  
        def \_\_len\_\_(self):  
            return math.ceil(len(self.x) / self.batch\_size)  
  
        def \_\_getitem\_\_(self, idx):  
            batch\_x = self.x[idx \* self.batch\_size:(idx + 1) \*  
            self.batch\_size]  
            batch\_y = self.y[idx \* self.batch\_size:(idx + 1) \*  
            self.batch\_size]  
  
            return np.array([  
                resize(imread(file\_name), (200, 200))  
                   for file\_name in batch\_x]), np.array(batch\_y)

## Methods

### \_\_getitem\_\_

\_\_getitem\_\_(index)

Gets batch at position index.

#### Arguments:

**index**: position of the batch in the Sequence.

#### Returns:

A batch

### \_\_iter\_\_

\_\_iter\_\_()

Create a generator that iterate over the Sequence.

### \_\_len\_\_

\_\_len\_\_()

Number of batch in the Sequence.

#### Returns:

The number of batches in the Sequence.

### on\_epoch\_end

on\_epoch\_end()

Method called at the end of every epoch.

# tf.keras.utils.SequenceEnqueuer

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer#top_of_page)

[Class SequenceEnqueuer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer#class_sequenceenqueuer)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/SequenceEnqueuer#methods)

## Class SequenceEnqueuer

Base class to enqueue inputs.

### Aliases:

Class tf.compat.v1.keras.utils.SequenceEnqueuer

Class tf.compat.v2.keras.utils.SequenceEnqueuer

Class tf.keras.utils.SequenceEnqueuer

Defined in [python/keras/utils/data\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/data_utils.py).

The task of an Enqueuer is to use parallelism to speed up preprocessing. This is done with processes or threads.

#### Example:

    enqueuer = SequenceEnqueuer(...)  
    enqueuer.start()  
    datas = enqueuer.get()  
    for data in datas:  
        # Use the inputs; training, evaluating, predicting.  
        # ... stop sometime.  
    enqueuer.close()

The enqueuer.get() should be an infinite stream of datas.

## \_\_init\_\_

\_\_init\_\_(  
    sequence,  
    use\_multiprocessing=False  
)

## Methods

### get

get()

Creates a generator to extract data from the queue.

Skip the data if it is None.

# Returns

Generator yielding tuples `(inputs, targets)`  
    or `(inputs, targets, sample\_weights)`.

### is\_running

is\_running()

### start

start(  
    workers=1,  
    max\_queue\_size=10  
)

Starts the handler's workers.

#### Arguments:

**workers**: Number of workers.

**max\_queue\_size**: queue size (when full, workers could block on put())

### stop

stop(timeout=None)

Stops running threads and wait for them to exit, if necessary.

Should be called by the same thread which called start().

#### Arguments:

**timeout**: maximum time to wait on thread.join()

tf.keras.utils.serialize\_keras\_object

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/serialize_keras_object#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/serialize_keras_object#aliases)

Aliases:

tf.compat.v1.keras.utils.serialize\_keras\_object

tf.compat.v2.keras.utils.serialize\_keras\_object

tf.keras.utils.serialize\_keras\_object

tf.keras.utils.serialize\_keras\_object(instance)

Defined in [python/keras/utils/generic\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/generic_utils.py).

# tf.keras.utils.to\_categorical

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/to_categorical#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/to_categorical#aliases)

[Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/utils/to_categorical#used_in_the_guide)

Converts a class vector (integers) to binary class matrix.

### Aliases:

tf.compat.v1.keras.utils.to\_categorical

tf.compat.v2.keras.utils.to\_categorical

tf.keras.utils.to\_categorical

tf.keras.utils.to\_categorical(  
    y,  
    num\_classes=None,  
    dtype='float32'  
)

Defined in [python/keras/utils/np\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/utils/np_utils.py).

### Used in the guide:

[The Keras Functional API in TensorFlow](https://www.tensorflow.org/beta/guide/keras/functional)

E.g. for use with categorical\_crossentropy.

#### Arguments:

**y**: class vector to be converted into a matrix (integers from 0 to num\_classes).

**num\_classes**: total number of classes.

**dtype**: The data type expected by the input. Default: 'float32'.

#### Returns:

A binary matrix representation of the input. The classes axis is placed last.

Module: tf.keras.wrappers

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers#top_of_page)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers#aliases)

[Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers#modules)

Wrappers for Keras models, providing compatibility with other frameworks.

Aliases:

Module tf.compat.v2.keras.wrappers

Module tf.keras.wrappers

Defined in [python/keras/api/\_v2/keras/wrappers/\_\_init\_\_.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/api/_v2/keras/wrappers/__init__.py).

Modules

[scikit\_learn](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn) module: Wrapper for using the Scikit-Learn API with Keras models.

# tf.keras.wrappers.scikit\_learn.KerasClassifier

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasClassifier#top_of_page)

[Class KerasClassifier](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasClassifier#class_kerasclassifier)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasClassifier#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasClassifier#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasClassifier#methods)

## Class KerasClassifier

Implementation of the scikit-learn classifier API for Keras.

### Aliases:

Class tf.compat.v1.keras.wrappers.scikit\_learn.KerasClassifier

Class tf.compat.v2.keras.wrappers.scikit\_learn.KerasClassifier

Class tf.keras.wrappers.scikit\_learn.KerasClassifier

Defined in [python/keras/wrappers/scikit\_learn.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/wrappers/scikit_learn.py).

## \_\_init\_\_

\_\_init\_\_(  
    build\_fn=None,  
    \*\*sk\_params  
)

## Methods

### check\_params

check\_params(params)

Checks for user typos in params.

#### Arguments:

**params**: dictionary; the parameters to be checked

#### Raises:

**ValueError**: if any member of params is not a valid argument.

### filter\_sk\_params

filter\_sk\_params(  
    fn,  
    override=None  
)

Filters sk\_params and returns those in fn's arguments.

#### Arguments:

**fn**: arbitrary function

**override**: dictionary, values to override sk\_params

#### Returns:

**res**: dictionary containing variables in both sk\_params and fn's arguments.

### fit

fit(  
    x,  
    y,  
    \*\*kwargs  
)

Constructs a new model with build\_fn & fit the model to (x, y).

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Training samples where n\_samples is the number of samples and n\_features is the number of features.

**y**: array-like, shape (n\_samples,) or (n\_samples, n\_outputs) True labels for x.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.fit

#### Returns:

**history**: object details about the training history at each epoch.

#### Raises:

**ValueError**: In case of invalid shape for y argument.

### get\_params

get\_params(\*\*params)

Gets parameters for this estimator.

#### Arguments:

**\*\*params**: ignored (exists for API compatibility).

#### Returns:

Dictionary of parameter names mapped to their values.

### predict

predict(  
    x,  
    \*\*kwargs  
)

Returns the class predictions for the given test data.

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Test samples where n\_samples is the number of samples and n\_features is the number of features.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.predict\_classes.

#### Returns:

**preds**: array-like, shape (n\_samples,) Class predictions.

### predict\_proba

predict\_proba(  
    x,  
    \*\*kwargs  
)

Returns class probability estimates for the given test data.

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Test samples where n\_samples is the number of samples and n\_features is the number of features.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.predict\_classes.

#### Returns:

**proba**: array-like, shape (n\_samples, n\_outputs) Class probability estimates. In the case of binary classification, to match the scikit-learn API, will return an array of shape (n\_samples, 2)(instead of (n\_sample, 1) as in Keras).

### score

score(  
    x,  
    y,  
    \*\*kwargs  
)

Returns the mean accuracy on the given test data and labels.

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Test samples where n\_samples is the number of samples and n\_features is the number of features.

**y**: array-like, shape (n\_samples,) or (n\_samples, n\_outputs) True labels for x.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.evaluate.

#### Returns:

**score**: float Mean accuracy of predictions on x wrt. y.

#### Raises:

**ValueError**: If the underlying model isn't configured to compute accuracy. You should pass metrics=["accuracy"] to the .compile() method of the model.

### set\_params

set\_params(\*\*params)

Sets the parameters of this estimator.

#### Arguments:

**\*\*params**: Dictionary of parameter names mapped to their values.

#### Returns:

self

# tf.keras.wrappers.scikit\_learn.KerasRegressor

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasRegressor#top_of_page)

[Class KerasRegressor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasRegressor#class_kerasregressor)

[Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasRegressor#aliases)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasRegressor#__init__)

[Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/wrappers/scikit_learn/KerasRegressor#methods)

## Class KerasRegressor

Implementation of the scikit-learn regressor API for Keras.

### Aliases:

Class tf.compat.v1.keras.wrappers.scikit\_learn.KerasRegressor

Class tf.compat.v2.keras.wrappers.scikit\_learn.KerasRegressor

Class tf.keras.wrappers.scikit\_learn.KerasRegressor

Defined in [python/keras/wrappers/scikit\_learn.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/wrappers/scikit_learn.py).

## \_\_init\_\_

\_\_init\_\_(  
    build\_fn=None,  
    \*\*sk\_params  
)

## Methods

### check\_params

check\_params(params)

Checks for user typos in params.

#### Arguments:

**params**: dictionary; the parameters to be checked

#### Raises:

**ValueError**: if any member of params is not a valid argument.

### filter\_sk\_params

filter\_sk\_params(  
    fn,  
    override=None  
)

Filters sk\_params and returns those in fn's arguments.

#### Arguments:

**fn**: arbitrary function

**override**: dictionary, values to override sk\_params

#### Returns:

**res**: dictionary containing variables in both sk\_params and fn's arguments.

### fit

fit(  
    x,  
    y,  
    \*\*kwargs  
)

Constructs a new model with build\_fn & fit the model to (x, y).

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Training samples where n\_samples is the number of samples and n\_features is the number of features.

**y**: array-like, shape (n\_samples,) or (n\_samples, n\_outputs) True labels for x.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.fit

#### Returns:

**history**: object details about the training history at each epoch.

### get\_params

get\_params(\*\*params)

Gets parameters for this estimator.

#### Arguments:

**\*\*params**: ignored (exists for API compatibility).

#### Returns:

Dictionary of parameter names mapped to their values.

### predict

predict(  
    x,  
    \*\*kwargs  
)

Returns predictions for the given test data.

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Test samples where n\_samples is the number of samples and n\_features is the number of features.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.predict.

#### Returns:

**preds**: array-like, shape (n\_samples,) Predictions.

### score

score(  
    x,  
    y,  
    \*\*kwargs  
)

Returns the mean loss on the given test data and labels.

#### Arguments:

**x**: array-like, shape (n\_samples, n\_features) Test samples where n\_samples is the number of samples and n\_features is the number of features.

**y**: array-like, shape (n\_samples,) True labels for x.

**\*\*kwargs**: dictionary arguments Legal arguments are the arguments of Sequential.evaluate.

#### Returns:

**score**: float Mean accuracy of predictions on x wrt. y.

### set\_params

set\_params(\*\*params)

Sets the parameters of this estimator.

#### Arguments:

**\*\*params**: Dictionary of parameter names mapped to their values.

#### Returns:

Self

# tf.compat.v1.keras.layers.CuDNNGRU

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#top_of_page)

[Class CuDNNGRU](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#class_cudnngru)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#__init__)

[Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#properties)

[cell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNGRU#cell)

## Class CuDNNGRU

Fast GRU implementation backed by cuDNN.

Defined in [python/keras/layers/cudnn\_recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/cudnn_recurrent.py).

More information about cuDNN can be found on the [NVIDIA developer website](https://developer.nvidia.com/cudnn). Can only be run on GPU.

#### Arguments:

**units**: Positive integer, dimensionality of the output space.

**kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.

**recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.

**bias\_initializer**: Initializer for the bias vector.

**kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.

**recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.

**bias\_regularizer**: Regularizer function applied to the bias vector.

**activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").

**kernel\_constraint**: Constraint function applied to the kernel weights matrix.

**recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.

**bias\_constraint**: Constraint function applied to the bias vector.

**return\_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence.

**return\_state**: Boolean. Whether to return the last state in addition to the output.

**go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.

**stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    \*\*kwargs  
)

## Properties

### cell

### states

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states

reset\_states(states=None)

# tf.compat.v1.keras.layers.CuDNNLSTM

[**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#top_of_page)

[Class CuDNNLSTM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#class_cudnnlstm)

[\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#__init__)

[Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#properties)

[cell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM#cell)

## Class CuDNNLSTM

Fast LSTM implementation backed by cuDNN.

Defined in [python/keras/layers/cudnn\_recurrent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/cudnn_recurrent.py).

More information about cuDNN can be found on the [NVIDIA developer website](https://developer.nvidia.com/cudnn). Can only be run on GPU.

#### Arguments:

**units**: Positive integer, dimensionality of the output space.

**kernel\_initializer**: Initializer for the kernel weights matrix, used for the linear transformation of the inputs.

**unit\_forget\_bias**: Boolean. If True, add 1 to the bias of the forget gate at initialization. Setting it to true will also force bias\_initializer="zeros". This is recommended in [Jozefowicz et al.](http://www.jmlr.org/proceedings/papers/v37/jozefowicz15.pdf)

**recurrent\_initializer**: Initializer for the recurrent\_kernel weights matrix, used for the linear transformation of the recurrent state.

**bias\_initializer**: Initializer for the bias vector.

**kernel\_regularizer**: Regularizer function applied to the kernel weights matrix.

**recurrent\_regularizer**: Regularizer function applied to the recurrent\_kernel weights matrix.

**bias\_regularizer**: Regularizer function applied to the bias vector.

**activity\_regularizer**: Regularizer function applied to the output of the layer (its "activation").

**kernel\_constraint**: Constraint function applied to the kernel weights matrix.

**recurrent\_constraint**: Constraint function applied to the recurrent\_kernel weights matrix.

**bias\_constraint**: Constraint function applied to the bias vector.

**return\_sequences**: Boolean. Whether to return the last output. in the output sequence, or the full sequence.

**return\_state**: Boolean. Whether to return the last state in addition to the output.

**go\_backwards**: Boolean (default False). If True, process the input sequence backwards and return the reversed sequence.

**stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

## \_\_init\_\_

\_\_init\_\_(  
    units,  
    kernel\_initializer='glorot\_uniform',  
    recurrent\_initializer='orthogonal',  
    bias\_initializer='zeros',  
    unit\_forget\_bias=True,  
    kernel\_regularizer=None,  
    recurrent\_regularizer=None,  
    bias\_regularizer=None,  
    activity\_regularizer=None,  
    kernel\_constraint=None,  
    recurrent\_constraint=None,  
    bias\_constraint=None,  
    return\_sequences=False,  
    return\_state=False,  
    go\_backwards=False,  
    stateful=False,  
    \*\*kwargs  
)

## Properties

### cell

### states

## Methods

### get\_initial\_state

get\_initial\_state(inputs)

### reset\_states