tf.keras.layers.Dropout

TensorFlow 1 version (/versions/r1.15/api_docs/python/tf/keras/layers/Dropout)

View source on GitHub (https://github.com/keras-team/keras/tree/v2.8.0/keras/layers/core/dr L125)

Applies Dropout to the input.

Inherits From: `Layer` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer), `Module` (https://www.tensorflow.org/api_docs/python/tf/Module)

View aliases

Compat aliases for migration

See Migration guide (https://www.tensorflow.org/guide/migrate) for more details.

`tf.compat.v1.keras.layers.Dropout` (https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout)

```
tf.keras.layers.Dropout(
    rate, noise_shape=None, seed=None, **kwargs
)
```

Used in the notebooks

Used in the guide	Used in the tutorials
• Effective Tensorflow 2 (https://www.tensorflow.org/guide/effective_tf2)	• Overfit and underfit (https://www.tensorflow.org/tutorials/keras/overfit_and_unde
• The Functional API (https://www.tensorflow.org/guide/keras/functional)	• Simple audio recognition: Recognizing keywords (https://www.tensorflow.org/tutorials/audio/simple_audio)
• Save and load Keras models (https://www.tensorflow.org/guide/keras/save_and_serialize)	• Deep Convolutional Generative Adversarial Network (https://www.tensorflow.org/tutorials/generative/dcgan)
• Transfer learning and fine-tuning (https://www.tensorflow.org/guide/keras/transfer_learning)	• Basic text classification (https://www.tensorflow.org/tutorials/keras/text_classificatio
• Migrate checkpoint saving (https://www.tensorflow.org/guide/migrate/checkpoint_saver)	• Create an Estimator from a Keras model (https://www.tensorflow.org/tutorials/estimator/keras_model

The Dropout layer randomly sets input units to 0 with a frequency of `rate` at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when `training` is set to True such that no values are dropped during inference. When using `model.fit`, `training` will be appropriately set to True automatically, and in other contexts, you can set the kwarg explicitly to True when calling the layer.

(This is in contrast to setting `trainable=False` for a Dropout layer. `trainable` does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

```
>>> tf.random.set_seed(0)
>>> layer = tf.keras.layers.Dropout(.2, input_shape=(2,))
>>> data = np.arange(10).reshape(5, 2).astype(np.float32)
>>> print(data)
[[0. 1.]
 [2. 3.]
 [4. 5.]
 [6. 7.]
 [8. 9.]]
>>> outputs = layer(data, training=True)
>>> print(outputs)
tf.Tensor(
[[ 0.    1.25]
 [ 2.5   3.75]
 [ 5.    6.25]
 [ 7.5   8.75]
 [10.    0.  ]], shape=(5, 2), dtype=float32)
```

Args	
`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).