Keras

About Keras

Getting started

Developer guides

Keras 3 API documentation

- Models API
- Layers API
- Callbacks API
- Ops API
- Optimizers
- Metrics
- Losses
- Data loading
- Built-in small datasets
- Keras Applications
  - Xception
  - EfficientNet B0 to B7
  - EfficientNetV2 B0 to B3 and S, M, L
  - ConvNeXt Tiny, Small, Base, Large, XLarge
  - VGG16 and VGG19
  - ResNet and ResNetV2
  - MobileNet, MobileNetV2, and MobileNetV3
  - DenseNet
  - NasNetLarge and NasNetMobile
  - InceptionV3
  - InceptionResNetV2
- Mixed precision
- Multi-device distribution
- RNG API
- Utilities
- KerasTuner
- KerasCV
- KerasNLP

Keras 2 API documentation

Code examples

KerasTuner: Hyperparameter Tuning

KerasCV: Computer Vision Workflows

MobileNet, MobileNetV2, and MobileNetV3

`MobileNet` function

[source]

```
keras.applications.MobileNet(
    input_shape=None,
    alpha=1.0,
    depth_multiplier=1,
    dropout=0.001,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    name=None,
)
```

Instantiates the MobileNet architecture.

Reference

- MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

This function returns a Keras image classification model, optionally loaded with weights pre-trained on ImageNet.

For image classification use cases, see this page for detailed examples.

For transfer learning use cases, make sure to read the guide to transfer learning & fine-tuning.

Note: each Keras Application expects a specific kind of input preprocessing. For MobileNet, call `keras.applications.mobilenet.preprocess_input` on your inputs before passing them to the model. `mobilenet.preprocess_input` will scale input pixels between -1 and 1.

Arguments

- **input_shape**: Optional shape tuple, only to be specified if `include_top` is `False` (otherwise the input shape has to be `(224, 224, 3)` (with `"channels_last"` data format) or `(3, 224, 224)` (with `"channels_first"` data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 32. E.g. `(200, 200, 3)` would be one valid value. Defaults to `None`. `input_shape` will be ignored if the `input_tensor` is provided.
- **alpha**: Controls the width of the network. This is known as the width multiplier in the MobileNet paper.
  - If `alpha < 1.0`, proportionally decreases the number of filters in each layer.
  - If `alpha > 1.0`, proportionally increases the number of filters in each layer.
  - If `alpha == 1`, default number of filters from the paper are used at each layer. Defaults to `1.0`.
- **depth_multiplier**: Depth multiplier for depthwise convolution. This is called the resolution multiplier in the MobileNet paper. Defaults to `1.0`.
- **dropout**: Dropout rate. Defaults to `0.001`.
- **include_top**: Boolean, whether to include the fully-connected layer at the top of the network. Defaults to `True`.
- **weights**: One of `None` (random initialization), `"imagenet"` (pre-training on ImageNet), or the path to the weights file to be loaded. Defaults to `"imagenet"`.
- **input_tensor**: Optional Keras tensor (i.e. output of `layers.Input()`) to use as image input for the model. `input_tensor` is useful for sharing inputs between multiple different networks. Defaults to `None`.
- **pooling**: Optional pooling mode for feature extraction when `include_top` is `False`.
  - `None` (default) means that the output of the model will be the 4D tensor output of the last convolutional block.

  - `avg` means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
  - `max` means that global max pooling will be applied.
- **classes**: Optional number of classes to classify images into, only to be specified if `include_top` is `True`, and if no `weights` argument is specified. Defaults to `1000`.
- **classifier_activation**: A `str` or callable. The activation function to use on the "top" layer. Ignored unless `include_top=True`. Set `classifier_activation=None` to return the logits of the "top" layer. When loading pretrained weights, `classifier_activation` can only be `None` or `"softmax"`.
- **name**: String, the name of the model.

Returns

A model instance.

`MobileNetV2` function

[source]

```
keras.applications.MobileNetV2(
    input_shape=None,
    alpha=1.0,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    name=None,
)
```

Instantiates the MobileNetV2 architecture.

MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance.

Reference

- MobileNetV2: Inverted Residuals and Linear Bottlenecks (CVPR 2018)

This function returns a Keras image classification model, optionally loaded with weights pre-trained on ImageNet.

For image classification use cases, see this page for detailed examples.

For transfer learning use cases, make sure to read the guide to transfer learning & fine-tuning.

Note: each Keras Application expects a specific kind of input preprocessing. For MobileNetV2, call `keras.applications.mobilenet_v2.preprocess_input` on your inputs before passing them to the model. `mobilenet_v2.preprocess_input` will scale input pixels between -1 and 1.

Arguments

- **input_shape**: Optional shape tuple, only to be specified if `include_top` is `False` (otherwise the input shape has to be `(224, 224, 3)` (with `"channels_last"` data format) or `(3, 224, 224)` (with `"channels_first"` data format). It should have exactly 3 inputs channels, and width and height should be no smaller than 32. E.g. `(200, 200, 3)` would be one valid value. Defaults to `None`. `input_shape` will be ignored if the `input_tensor` is provided.
- **alpha**: Controls the width of the network. This is known as the width multiplier in the MobileNet paper.
  - If `alpha < 1.0`, proportionally decreases the number of filters in each layer.
  - If `alpha > 1.0`, proportionally increases the number of filters in each layer.
  - If `alpha == 1`, default number of filters from the paper are used at each layer. Defaults to `1.0`.
- **include_top**: Boolean, whether to include the fully-connected layer at the top of the network. Defaults to `True`.
- **weights**: One of `None` (random initialization), `"imagenet"` (pre-training on ImageNet), or the path to the weights file to be loaded. Defaults to `"imagenet"`.

- **input_tensor**: Optional Keras tensor (i.e. output of `layers.Input()`) to use as image input for the model. `input_tensor` is useful for sharing inputs between multiple different networks. Defaults to `None`.
- **pooling**: Optional pooling mode for feature extraction when `include_top` is `False`.
    - `None` (default) means that the output of the model will be the 4D tensor output of the last convolutional block.
    - `avg` means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
    - `max` means that global max pooling will be applied.
- **classes**: Optional number of classes to classify images into, only to be specified if `include_top` is `True`, and if no `weights` argument is specified. Defaults to `1000`.
- **classifier_activation**: A `str` or callable. The activation function to use on the "top" layer. Ignored unless `include_top=True`. Set `classifier_activation=None` to return the logits of the "top" layer. When loading pretrained weights, `classifier_activation` can only be `None` or `"softmax"`.
- **name**: String, the name of the model.

Returns

A model instance.

`MobileNetV3Small` function

[source]

```
keras.applications.MobileNetV3Small(
    input_shape=None,
    alpha=1.0,
    minimalistic=False,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    classes=1000,
    pooling=None,
    dropout_rate=0.2,
    classifier_activation="softmax",
    include_preprocessing=True,
    name="MobileNetV3Small",
)
```

Instantiates the MobileNetV3Small architecture.

Reference

- Searching for MobileNetV3 (ICCV 2019)

The following table describes the performance of MobileNets v3:

MACs stands for Multiply Adds

Classification Checkpoint	MACs(M)	Parameters(M)	Top1 Accuracy	Pixel1 CPU(ms)
mobilenet_v3_large_1.0_224	217	5.4	75.6	51.2
mobilenet_v3_large_0.75_224	155	4.0	73.3	39.8
mobilenet_v3_large_minimalistic_1.0_224	209	3.9	72.3	44.1
mobilenet_v3_small_1.0_224	66	2.9	68.1	15.8
mobilenet_v3_small_0.75_224	44	2.4	65.4	12.8
mobilenet_v3_small_minimalistic_1.0_224	65	2.0	61.9	12.2

For image classification use cases, see this page for detailed examples.

For transfer learning use cases, make sure to read the guide to transfer learning & fine-tuning.

Note: each Keras Application expects a specific kind of input preprocessing. For MobileNetV3, by default input preprocessing is included as a part of the model (as a `Rescaling` layer), and thus `keras.applications.mobilenet_v3.preprocess_input` is actually a pass-through function. In this use case, MobileNetV3 models expect their inputs to be float tensors of pixels with values in the `[0-255]` range. At the same time, preprocessing as a part of the model (i.e. `Rescaling` layer) can be disabled by setting `include_preprocessing` argument to `False`. With preprocessing disabled MobileNetV3 models expect their inputs to be float tensors of pixels with values in the `[-1, 1]` range.

Arguments

- **input_shape**: Optional shape tuple, to be specified if you would like to use a model with an input image resolution that is not `(224, 224, 3)`. It should have exactly 3 inputs channels. You can also omit this option if you would like to infer input_shape from an input_tensor. If you choose to include both input_tensor and input_shape then input_shape will be used if they match, if the shapes do not match then we will throw an error. E.g. `(160, 160, 3)` would be one valid value.
- **alpha**: controls the width of the network. This is known as the depth multiplier in the MobileNetV3 paper, but the name is kept for consistency with MobileNetV1 in Keras.
  - If `alpha < 1.0`, proportionally decreases the number of filters in each layer.
  - If `alpha > 1.0`, proportionally increases the number of filters in each layer.
  - If `alpha == 1`, default number of filters from the paper are used at each layer.
- **minimalistic**: In addition to large and small models this module also contains so-called minimalistic models, these models have the same per-layer dimensions characteristic as MobilenetV3 however, they don't utilize any of the advanced blocks (squeeze-and-excite units, hard-swish, and 5x5 convolutions). While these models are less efficient on CPU, they are much more performant on GPU/DSP.
- **include_top**: Boolean, whether to include the fully-connected layer at the top of the network. Defaults to `True`.
- **weights**: String, one of `None` (random initialization), `"imagenet"` (pre-training on ImageNet), or the path to the weights file to be loaded.
- **input_tensor**: Optional Keras tensor (i.e. output of `layers.Input()`) to use as image input for the model.
- **pooling**: String, optional pooling mode for feature extraction when `include_top` is `False`.
  - `None` means that the output of the model will be the 4D tensor output of the last convolutional block.
  - `avg` means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
  - `max` means that global max pooling will be applied.
- **classes**: Integer, optional number of classes to classify images into, only to be specified if `include_top` is `True`, and if no `weights` argument is specified.
- **dropout_rate**: fraction of the input units to drop on the last layer.
- **classifier_activation**: A `str` or callable. The activation function to use on the "top" layer. Ignored unless `include_top=True`. Set `classifier_activation=None` to return the logits of the "top" layer. When loading pretrained weights, `classifier_activation` can only be `None` or `"softmax"`.
- **include_preprocessing**: Boolean, whether to include the preprocessing layer (`Rescaling`) at the bottom of the network. Defaults to `True`.
- **name**: String, the name of the model.

Call arguments

- **inputs**: A floating point `numpy.array` or backend-native tensor, 4D with 3 color channels, with values in the range `[0, 255]` if `include_preprocessing` is `True` and in the range `[-1, 1]` otherwise.

Returns

A model instance.

[source]

`MobileNetV3Large` function

```
keras.applications.MobileNetV3Large(
    input_shape=None,
    alpha=1.0,
    minimalistic=False,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    classes=1000,
    pooling=None,
    dropout_rate=0.2,
    classifier_activation="softmax",
    include_preprocessing=True,
    name="MobileNetV3Large",
)
```

Instantiates the MobileNetV3Large architecture.

Reference

- Searching for MobileNetV3 (ICCV 2019)

The following table describes the performance of MobileNets v3:

MACs stands for Multiply Adds

Classification Checkpoint	MACs(M)	Parameters(M)	Top1 Accuracy	Pixel1 CPU(ms)
mobilenet_v3_large_1.0_224	217	5.4	75.6	51.2
mobilenet_v3_large_0.75_224	155	4.0	73.3	39.8
mobilenet_v3_large_minimalistic_1.0_224	209	3.9	72.3	44.1
mobilenet_v3_small_1.0_224	66	2.9	68.1	15.8
mobilenet_v3_small_0.75_224	44	2.4	65.4	12.8
mobilenet_v3_small_minimalistic_1.0_224	65	2.0	61.9	12.2

For image classification use cases, see this page for detailed examples.

For transfer learning use cases, make sure to read the guide to transfer learning & fine-tuning.

Note: each Keras Application expects a specific kind of input preprocessing. For MobileNetV3, by default input preprocessing is included as a part of the model (as a `Rescaling` layer), and thus `keras.applications.mobilenet_v3.preprocess_input` is actually a pass-through function. In this use case, MobileNetV3 models expect their inputs to be float tensors of pixels with values in the `[0-255]` range. At the same time, preprocessing as a part of the model (i.e. `Rescaling` layer) can be disabled by setting `include_preprocessing` argument to `False`. With preprocessing disabled MobileNetV3 models expect their inputs to be float tensors of pixels with values in the `[-1, 1]` range.

Arguments

- **input_shape**: Optional shape tuple, to be specified if you would like to use a model with an input image resolution that is not `(224, 224, 3)`. It should have exactly 3 inputs channels. You can also omit this option if you would like to infer input_shape from an input_tensor. If you choose to include both input_tensor and input_shape then input_shape will be used if they match, if the shapes do not match then we will throw an error. E.g. `(160, 160, 3)` would be one valid value.
- **alpha**: controls the width of the network. This is known as the depth multiplier in the MobileNetV3 paper, but the name is kept for consistency with MobileNetV1 in Keras.
  - If `alpha < 1.0`, proportionally decreases the number of filters in each layer.
  - If `alpha > 1.0`, proportionally increases the number of filters in each layer.
  - If `alpha == 1`, default number of filters from the paper are used at each layer.
- **minimalistic**: In addition to large and small models this module also contains so-called minimalistic models, these models have the same per-layer dimensions characteristic as MobilenetV3 however, they don't utilize any of the advanced blocks (squeeze-and-excite units, hard-swish, and 5x5 convolutions). While these models are less efficient on CPU, they are much more performant on GPU/DSP.

- **include_top**: Boolean, whether to include the fully-connected layer at the top of the network. Defaults to `True`.
- **weights**: String, one of `None` (random initialization), `"imagenet"` (pre-training on ImageNet), or the path to the weights file to be loaded.
- **input_tensor**: Optional Keras tensor (i.e. output of `layers.Input()`) to use as image input for the model.
- **pooling**: String, optional pooling mode for feature extraction when `include_top` is `False`.
  - `None` means that the output of the model will be the 4D tensor output of the last convolutional block.
  - `avg` means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.
  - `max` means that global max pooling will be applied.
- **classes**: Integer, optional number of classes to classify images into, only to be specified if `include_top` is `True`, and if no `weights` argument is specified.
- **dropout_rate**: fraction of the input units to drop on the last layer.
- **classifier_activation**: A `str` or callable. The activation function to use on the "top" layer. Ignored unless `include_top=True`. Set `classifier_activation=None` to return the logits of the "top" layer. When loading pretrained weights, `classifier_activation` can only be `None` or `"softmax"`.
- **include_preprocessing**: Boolean, whether to include the preprocessing layer (`Rescaling`) at the bottom of the network. Defaults to `True`.
- **name**: String, the name of the model.

Call arguments

- **inputs**: A floating point `numpy.array` or backend-native tensor, 4D with 3 color channels, with values in the range `[0, 255]` if `include_preprocessing` is `True` and in the range `[-1, 1]` otherwise.

Returns

A model instance.