Applications

Kerasの応用は事前学習した重みを利用可能な深層学習のモデルです． これらのモデルは予測，特徴量抽出そしてfine-tuningのために利用できます．

モデルをインスタンス化すると重みは自動的にダウンロードされます．重みは`~/.keras/models/`に格納されます．

利用可能なモデル

ImageNetで学習した重みをもつ画像分類のモデル:

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- InceptionResNetV2
- MobileNet
- DenseNet
- NASNet
- MobileNetV2

これら全てのアーキテクチャは全てのバックエンド（TensorFlowやTheano，CNTK）と互換性があり，モデルはインスタンス化する時はKerasの設定ファイル`~/.keras/keras.json`に従って画像のデータフォーマットが設定されます． 例えば，`image_dim_ordering=channels_last`とした際は，このリポジトリからロードされるモデルは，TensorFlowの次元の順序"Height-Width-Depth"にしたがって構築されます．

注意：

- `Keras < 2.2.0`ではXceptionモデルはTensorFlowでのみ利用可能です．これは`SeparableConvolution`レイヤーに依存しているからです．
- `Keras < 2.1.5`ではMobileNetモデルはTensorFlowでのみ利用可能です．これは`DepthwiseConvolution`レイヤーに依存しているからです．

画像分類モデルの使用例

Classify ImageNet classes with ResNet50

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.82658225), (u'n01871265', u'tusker', 0.11
```

Extract features with VGG16

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np

model = VGG16(weights='imagenet', include_top=False)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

Extract features from an arbitrary intermediate layer with VGG19

```
from keras.applications.vgg19 import VGG19
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input
from keras.models import Model
import numpy as np

base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('block4_pool').output)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

block4_pool_features = model.predict(x)
```

Fine-tune InceptionV3 on a new set of classes

```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K

# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)
# and a logistic layer -- let's say we have 200 classes
predictions = Dense(200, activation='softmax')(x)

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)

# first: train only the top layers (which were randomly initialized)
# i.e. freeze all convolutional InceptionV3 layers
for layer in base_model.layers:
    layer.trainable = False

# compile the model (should be done *after* setting layers to non-trainable)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')

# train the model on the new data for a few epochs
model.fit_generator(...)

# at this point, the top layers are well trained and we can start fine-tuning
# convolutional layers from inception V3. We will freeze the bottom N layers
# and train the remaining top layers.

# let's visualize layer names and layer indices to see how many layers
# we should freeze:
for i, layer in enumerate(base_model.layers):
   print(i, layer.name)

# we chose to train the top 2 inception blocks, i.e. we will freeze
# the first 249 layers and unfreeze the rest:
for layer in model.layers[:249]:
   layer.trainable = False
for layer in model.layers[249:]:
   layer.trainable = True

# we need to recompile the model for these modifications to take effect
# we use SGD with a low learning rate
from keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy')

# we train our model again (this time fine-tuning the top 2 inception blocks
# alongside the top Dense layers
model.fit_generator(...)
```

Build InceptionV3 over a custom input tensor

```
from keras.applications.inception_v3 import InceptionV3
from keras.layers import Input

# this could also be the output a different Keras model or layer
input_tensor = Input(shape=(224, 224, 3))  # this assumes K.image_data_format() == 'channels

model = InceptionV3(input_tensor=input_tensor, weights='imagenet', include_top=True)
```

Documentation for individual models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

トップ1とトップ5の精度はImageNetの検証データセットを参照しています.

Xception

```
keras.applications.xception.Xception(include_top=True, weights='imagenet', input_tensor=None
```

ImageNetで事前学習した重みを利用可能なXception V1モデル.

ImageNetにおいて，このモデルのtop-1のvalidation accuracyは0.790で，top-5のvalidation accuracyは0.945です.

データフォーマットは `'channels_last'` (height, width, channels)のみサポートしています.

デフォルトの入力サイズは299x299.

引数

- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None` (ランダム初期化) か `'imagenet'` (ImageNetで学習した重み) のどちらか一方.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり，`layers.Input()` の出力)
- input_shape: オプショナルなshapeのタプル，`include_top` がFalseの場合のみ指定可能 (そうでないときは入力のshapeは `(299, 299, 3)`). 正確に3つの入力チャンネルをもつ必要があり，width と height は71以上にする必要があります. 例えば `(150, 150, 3)` は有効な値です.

- pooling: 特徴量抽出のためのオプショナルなpooling mode, `include_top` が `False` の場合のみ指定可能.
  - `None` : モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'` : 最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています.
  - `'max'` : global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数, `include_top` が `True` かつ `weights` が指定されていない場合のみ指定可能.

戻り値

Kerasの `Model` インスタンス.

参考文献

- Xception: Deep Learning with Depthwise Separable Convolutions

ライセンス

この重みは私達自身が学習したもので，MITライセンスの下で公開されています.

VGG16

```
keras.applications.vgg16.VGG16(include_top=True, weights='imagenet', input_tensor=None, inpu
```

ImageNetで事前学習した重みを利用可能なVGG16モデル.

`'channels_first'` データフォーマット (channels, height, width) か `'channels_last'` データフォーマット (height, width, channels)の両方で構築可能です.

デフォルトの入力サイズは224x224.

引数

- include_top: ネットワークの出力層側にある3つの全結合層を含むかどうか.
- weights: `None` (ランダム初期化) か `'imagenet'` (ImageNetで学習した重み) のどちらか一方.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり，`layers.Input()` の出力)
- input_shape: オプショナルなshapeのタプル，`include_top` が `False` の場合のみ指定可能 (そうでないときは入力のshapeは `(224, 224, 3)` (`'channels_last'` データフォーマットのとき) か `(3, 224, 224)` (`'channels_first'` データフォーマットのとき)). 正確に3つの入力チャンネルを

もつ必要があり，width と height は48以上にする必要があります．例えば `(200, 200, 3)` は有効値．

- pooling: 特徴量抽出のためのオプショナルなpooling mode，`include_top` が `False` の場合のみ指定可能．
  - `None` ：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています．
  - `'avg'` ：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています．
  - `'max'` ：global max poolingが適用されることを意味します．
- classes: 画像のクラス分類のためのオプショナルなクラス数，`include_top` が `True` かつ `weights` が指定されていない場合のみ指定可能．

戻り値

Kerasの `Model` インスタンス．

参考文献

- Very Deep Convolutional Networks for Large-Scale Image Recognition: please cite this paper if you use the VGG models in your work.

ライセンス

この重みはOxford大学のVGGによりCreative Commons Attribution Licenseの下で公開されたものを移植しています．

VGG19

```
keras.applications.vgg19.VGG19(include_top=True, weights='imagenet', input_tensor=None, inpu
```

ImageNetで事前学習した重みを利用可能なVGG19モデル．

`'channels_first'` データフォーマット (channels, height, width) か `'channels_last'` データフォーマット (height, width, channels)の両方で構築可能です．

デフォルトの入力サイズは224x224．

引数

- include_top: ネットワークの出力層側にある3つの全結合層を含むかどうか．
- weights: `None` (ランダム初期化) か `'imagenet'` (ImageNetで学習した重み) の一方．
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり，`layers.Input()` の出力)

- input_shape: オプショナルなshapeのタプル，`include_top`がFalseの場合のみ指定可能 (そうでないときは入力のshapeは`(224, 224, 3)` (`'channels_last'`データフォーマットのとき) か`(3, 224, 224)` (`'channels_first'`データフォーマットのとき)). 正確に3つの入力チャンネルをもつ必要があり，width と height は48以上にする必要があります．例えば`(200, 200, 3)`は有効値.
- pooling: 特徴量抽出のためのオプショナルなpooling mode，`include_top`が`False`の場合のみ指定可能.
  - `None`：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'`：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています.
  - `'max'`：global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数，`include_top`が`True`かつ`weights`が指定されていない場合のみ指定可能.

戻り値

Kerasの`Model`インスタンス.

参考文献

- Very Deep Convolutional Networks for Large-Scale Image Recognition

ライセンス

この重みはOxford大学のVGGによりCreative Commons Attribution Licenseの下で公開されたものを移植しています.

ResNet50

```
keras.applications.resnet50.ResNet50(include_top=True, weights='imagenet', input_tensor=None
```

ImageNetで事前学習した重みを利用可能なResNet50モデル.

`'channels_first'`データフォーマット (channels, height, width) か`'channels_last'`データフォーマット (height, width, channels)の両方で構築可能です.

デフォルトの入力サイズは224x224.

引数

- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None` (ランダム初期化) か`'imagenet'` (ImageNetで学習した重み) の一方.

- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり，`layers.Input()`の出力)
- input_shape: オプショナルなshapeのタプル，`include_top`がFalseの場合のみ指定可能 (そうでないときは入力のshapeは`(224, 224, 3)` (`'channels_last'`データフォーマットのとき) か`(3, 224, 224)` (`'channels_first'`データフォーマットのとき)). 正確に3つの入力チャンネルをもつ必要があり，width と height は197以上にする必要があります．例えば`(200, 200, 3)`は有効値．
- pooling: 特徴量抽出のためのオプショナルなpooling mode，`include_top`が`False`の場合のみ指定可能．
  - `None`：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています．
  - `'avg'`：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています．
  - `'max'`：global max poolingが適用されることを意味します．
- classes: 画像のクラス分類のためのオプショナルなクラス数，`include_top`が`True`かつ`weights`が指定されていない場合のみ指定可能．

戻り値

Kerasの`Model`インスタンス．

参考文献

- Deep Residual Learning for Image Recognition

ライセンス

この重みはKaiming HeによりMITライセンスの下で公開されたものを移植しています．

InceptionV3

```
keras.applications.inception_v3.InceptionV3(include_top=True, weights='imagenet', input_tens
```

ImageNetで事前学習した重みを利用可能なInception V3モデル．

`'channels_first'`データフォーマット (channels, height, width) か`'channels_last'`データフォーマット (height, width, channels)の両方で構築可能です．

デフォルトの入力サイズは299x299．

引数

- include_top: ネットワークの出力層側にある全結合層を含むかどうか．

- weights: `None` (ランダム初期化) か `'imagenet'` (ImageNetで学習した重み) の一方.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり, `layers.Input()` の出力)
- input_shape: オプショナルなshapeのタプル, `include_top` がFalseの場合のみ指定可能 (そうでないときは入力のshapeは `(299, 299, 3)` (`'channels_last'` データフォーマットのとき) か `(3, 299, 299)` (`'channels_first'` データフォーマットのとき)). 正確に3つの入力チャンネルをもつ必要があり, width と height は139以上にする必要があります. 例えば `(150, 150, 3)` は有効値.
- pooling: 特徴量抽出のためのオプショナルなpooling mode, `include_top` が `False` の場合のみ指定可能.
  - `None` : モデルの出力が, 最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'` : 最後のconvolutional layerの出力にglobal average poolingが適用されることで, モデルの出力が2階テンソルになることを意味しています.
  - `'max'` : global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数, `include_top` が `True` かつ `weights` が指定されていない場合のみ指定可能.

戻り値

Kerasの `Model` インスタンス.

参考文献

- Rethinking the Inception Architecture for Computer Vision

ライセンス

この重みは Apacheライセンスの下で公開されています.

InceptionResNetV2

```
keras.applications.inception_resnet_v2.InceptionResNetV2(include_top=True, weights='imagenet
```

ImageNetで事前学習したInception-ResNet V2モデル.

`'channels_first'` データフォーマット (channels, height, width) か `'channels_last'` データフォーマット (height, width, channels)の両方で構築可能です.

デフォルトの入力サイズは299x299.

引数

- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None` (ランダム初期化) か `'imagenet'` (ImageNetで学習した重み) の一方.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり, `layers.Input()` の出力)
- input_shape: オプショナルなshapeのタプル, `include_top` がFalseの場合のみ指定可能 (そうでないときは入力のshapeは `(299, 299, 3)` (`'channels_last'` データフォーマットのとき) か `(3, 299, 299)` (`'channels_first'` データフォーマットのとき)). 正確に3つの入力チャンネルをもつ必要があり, width と height は139以上にする必要があります. 例えば `(150, 150, 3)` は有効値.
- pooling: 特徴量抽出のためのオプショナルなpooling mode, `include_top` が `False` の場合のみ指定可能.
  - `None` : モデルの出力が, 最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'` : 最後のconvolutional layerの出力にglobal average poolingが適用されることで, モデルの出力が2階テンソルになることを意味しています.
  - `'max'` : global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数, `include_top` が `True` かつ `weights` が指定されていない場合のみ指定可能.

戻り値

Kerasのモデルインスタンス.

参考文献

- Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning

ライセンス

この重みは Apacheライセンスの下で公開されています.

MobileNet

```
keras.applications.mobilenet.MobileNet(input_shape=None, alpha=1.0, depth_multiplier=1, drop
```

ImageNetで事前学習したMobileNetモデル.

データフォーマットが `'channels_last'` (height, width, channels)の時のみサポートされることに注意してください.

`load_model` からMobileNetモデルをロードするには, カスタムオブジェクトの `relu6` をインポートし, `custom_objects` パラメータに渡してください.

例

```
model = load_model('mobilenet.h5', custom_objects={
                   'relu6': mobilenet.relu6})
```

デフォルトの入力サイズは224x224.

引数

- input_shape: オプショナルなshapeのタプル， `include_top` が `False` の場合のみ指定可能 (そうでないときは入力のshapeは `(224, 224, 3)` (`'channels_last'` データフォーマットのとき) か `(3, 224, 224)` (`'channels_first'` データフォーマットのとき)). 正確に3つの入力チャンネルをもつ必要があり，width と height は32以上にする必要があります．例えば `(200, 200, 3)` は有効値.
- alpha: ネットワークの幅の制御.
  - `alpha` < 1.0の場合，各レイヤーのフィルタ数を比例して減少させます.
  - `alpha` > 1.0の場合，各レイヤーのフィルタ層を比例して増加させます.
  - `alpha` = 1の場合，論文のデフォルトのフィルタ数が各レイヤーで使われます.
- depth_multiplier: 深さ方向の畳み込みための深さ乗数（resolution multiplierとも呼ばれます）
- dropout: ドロップアウト率
- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None` (ランダム初期化) か `'imagenet'` (ImageNetで学習した重み) の一方.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり， `layers.Input()` の出力)
- pooling: 特徴量抽出のためのオプショナルなpooling mode， `include_top` が `False` の場合のみ指定可能.
  - `None` ：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'` ：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています.
  - `'max'` ：global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数， `include_top` が `True` かつ `weights` が指定されていない場合のみ指定可能.

戻り値

Kerasの `Model` インスタンス.

参考文献

- MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

ライセンス

この重みは Apacheライセンスの下で公開されています.

DenseNet

```
keras.applications.densenet.DenseNet121(include_top=True, weights='imagenet', input_tensor=N
keras.applications.densenet.DenseNet169(include_top=True, weights='imagenet', input_tensor=N
keras.applications.densenet.DenseNet201(include_top=True, weights='imagenet', input_tensor=N
```

ImageNetで事前学習したDenseNetモデル.

このモデルは`'channels_first'`データフォーマット(channels, height, width)と`'channels_last'`データフォーマット(height, width, channels)の両方で構築可能です.

デフォルトの入力サイズは224x224.

引数

- blocks: 4つのdenseレイヤーために構築するブロックの個数.
- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None`（ランダム初期化），'imagenet'（ImageNetでの事前学習），ロードする重みのファイルへのパスのいずれか.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル（つまり，`layers.Input()`の出力）
- input_shape: オプショナルなshapeのタプル，`include_top`が`False`の場合のみ指定可能 (そうでないときは入力のshapeは`(224, 224, 3)`(`'channels_last'`データフォーマットのとき) か`(3, 224, 224)`(`'channels_first'`データフォーマットのとき)). 正確に3つの入力チャンネルをもつ必要があります.
- pooling: 特徴量抽出のためのオプショナルなpooling mode，`include_top`が`False`の場合のみ指定可能.
  - `None`：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'`：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています.
  - `'max'`：global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数，`include_top`が`True`かつ`weights`が指定されていない場合のみ指定可能.

戻り値

Kerasのモデルインスタンス.

参考文献

- Densely Connected Convolutional Networks (CVPR 2017 Best Paper Award)

ライセンス

この重みは三条項BSDライセンスの下で公開されています.

NASNet

```
keras.applications.nasnet.NASNetLarge(input_shape=None, include_top=True, weights='imagenet'
keras.applications.nasnet.NASNetMobile(input_shape=None, include_top=True, weights='imagenet
```

ImageNetで事前学習したNeural Architecture Search Network (NASNet)モデル.

デフォルトの入力サイズは，NASNetLargeモデルは331x331，NASNetMobileモデルは224x224.

引数

- input_shape: オプショナルなshapeのタプル，`include_top`が`False`の場合のみ指定可能（そうでないときの入力のshapeは，NASNetMobileなら`(224, 224, 3)`（`'channels_last'`データフォーマットのとき）または`(3, 224, 224)`（`'channels_first'`データフォーマットのとき），NASNetLargeなら`(331, 331, 3)`（`'channels_last'`データフォーマットのとき）または`(3, 331, 331)`（`'channels_first'`データフォーマットのとき））．正確に3つの入力チャンネルをもつ必要があり，width と height は32以上にする必要があります．例えば`(200, 200, 3)`は有効値.
- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None`（ランダム初期化）か`'imagenet'`（ImageNetで学習した重み）の一方.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル（つまり，`layers.Input()`の出力）
- pooling: 特徴量抽出のためのオプショナルなpooling mode，`include_top`が`False`の場合のみ指定可能.
  - `None`：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています.
  - `'avg'`：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています.
  - `'max'`：global max poolingが適用されることを意味します.
- classes: 画像のクラス分類のためのオプショナルなクラス数，`include_top`が`True`かつ`weights`が指定されていない場合のみ指定可能.

戻り値

Kerasの`Model`インスタンス.

参考文献

- Learning Transferable Architectures for Scalable Image Recognition

ライセンス

この重みはApacheライセンスの下で公開されています.

MobileNetV2

```
keras.applications.mobilenetv2.MobileNetV2(input_shape=None, alpha=1.0, depth_multiplier=1,
```

ImageNetで事前学習したMobileNetV2モデル.

データフォーマットが `'channels_last'` (height, width, channels)の時のみサポートされることに注意してください.

`load_model` からMobileNetV2モデルをロードするには，カスタムオブジェクトの `relu6` をインポートし，`custom_objects` パラメータに渡してください.

例

```
model = load_model('mobilenet_v2.h5', custom_objects={
                   'relu6': mobilenetv2.relu6})
```

デフォルトの入力サイズは224x224.

引数

- input_shape: オプショナルなshapeのタプル，入力画像の解像度が(224, 224, 3)でないときは指定すべきです． (224, 224, 3)のように正確に3つの入力チャネルが必要です．input_tensorからinput_shapeが推論できるならこのオプションは省くこともできます．入力するinput_tensorとinput_shapeを決めてそれらの値がマッチしていればinput_shapeが用いられ，shapeがマッチしなければエラーを送出します．例えば `(160, 160, 3)` は妥当な値です.
- alpha: ネットワークの幅の制御． MobileNetV2の論文ではwidth multiplierとして知られています.
  - `alpha` < 1.0の場合，各レイヤーのフィルタ数を比例して減少させます.
  - `alpha` > 1.0の場合，各レイヤーのフィルタ層を比例して増加させます.
  - `alpha` = 1の場合，論文のデフォルトのフィルタ数が各レイヤーで使われます.
- depth_multiplier: 深さ方向の畳み込みための深さ乗数（resolution multiplierとも呼ばれます）
- include_top: ネットワークの出力層側にある全結合層を含むかどうか.
- weights: `None` (ランダム初期化)か，`'imagenet'` (ImageNetで学習した重み)か，ロードする重みファイルへのパスのいずれか.
- input_tensor: モデルの入力画像として利用するためのオプションのKerasテンソル (つまり，`layers.Input()` の出力)
- pooling: 特徴量抽出のためのオプショナルなpooling mode，`include_top` が `False` の場合のみ指定可能.
  - `None` ：モデルの出力が，最後のconvolutional layerの4階テンソルであることを意味しています.

    - `'avg'`：最後のconvolutional layerの出力にglobal average poolingが適用されることで，モデルの出力が2階テンソルになることを意味しています．
    - `'max'`：global max poolingが適用されることを意味します．
- classes: 画像のクラス分類のためのオプショナルなクラス数，`include_top`がTrueかつ`weights`が指定されていない場合のみ指定可能．

戻り値

Kerasのモデルインスタンス．

参考文献

- MobileNetV2: Inverted Residuals and Linear Bottlenecks

ライセンス

この重みはApacheライセンスの下で公開されています．