Image Classification

by Sanjeev Gupta

Solve an image classification problem on the cats-vs-dogs dataset by training 'mini-Exception-like' model based on the instructions given below:

Set the global random seed to 42

We are using a cat-vs-dogs dataset here. You will have to download it using instruction.

Download the data through the following command in your notebook

!git clone -b master https://github.com/sumanthk94/DA_225-o.git

Use the image_dataset_from_directory utility from tensorflow.keras.utils to make appropriate datasets. (0)

Building the Model based on this model_summary and its corresponding model_plot. Ensure that you follow the trailing instructions:(16)

i). For the initial layers of model mentioned in this summary, random flip (horizontal), random rotation of 0.1, random zoom of 0.2, rescaling by 1./255, and set kernel_size = 5 and use_bias=False in the convolution layer.

ii). Define a block of following layers:

* Batch Normalization layer
* Activation layer with relu as activation function
* Depth wise separable layer (kernel size = 3)
* Batch Normalization layer
* Activation layer with relu as activation function
* Depth wise separable layer (kernel size = 5)
* Batch Normalization layer
* Activation layer with relu as activation function
* Depth wise separable layer (kernel size = 7)
* Maxpool2D layer (poolsize =3, stride=2)
* Convolution layer
* 'Add layer' due to a residual connection. Infer connection points of the skip connection from the model summary and model plot.

Infer unspecified arguments from the summary

iii). The block defined in (ii) repeats 4 times. Note that in each repitition, the number of filters changes. Infer this from the model plot/summary.

iv). The last two layers are GlobalAveragePooling and Dense layers. The dense layer is the output layer (infer the number of neurons and the activation function).

Compile model with rmsprop as an optimizer with appropriate loss and metric for this respective problem. (1)

Fit the model with a batch_size of 32 for 20 epochs. (Don't use EarlyStopping callback). Use the validation dataset from the data you downloaded. We have specified a small no. of epochs because training may take time. Try running colab on GPU by going to Edit > Notebook accelerator > Hardware Accelerator > GPU. (1)

Return the history as a DataFrame. Show loss and accuracy for training and validation through appropriate plots.(1)

Evaluate the Model on test dataset from the data you downloaded.(1)

Note:

If you are using any parameter values or arguments apart from the ones mentioned or the ones that you must infer, state explicitly where and why you are using them.

Also verify that the total no. of params of your model are the same as that mentioned in model_summary txt file given to you

Image Classification

by Sanjeev Gupta

Solve an image classification problem on the cats-vs-dogs dataset by training 'mini-Exception-like' model based on the instructions given below:

Set the global random seed to 42

We are using a cat-vs-dogs dataset here. You will have to download it using instruction.

Download the data through the following command in your notebook

!git clone -b master https://github.com/sumanthk94/DA_225-o.git

Use the image_dataset_from_directory utility from tensorflow.keras.utils to make appropriate datasets. (0)

Building the Model based on this model_summary and its corresponding model_plot. Ensure that you follow the trailing instructions:(16)

i). For the initial layers of model mentioned in this summary, random flip (horizontal), random rotation of 0.1, random zoom of 0.2, rescaling by 1./255, and set kernel_size = 5 and use_bias=False in the convolution layer.

ii). Define a block of following layers:

- Batch Normalization layer
- Activation layer with relu as activation function
- Depth wise separable layer (kernel size = 3)
- Batch Normalization layer
- Activation layer with relu as activation function
- Depth wise separable layer (kernel size = 5)
- Batch Normalization layer
- Activation layer with relu as activation function
- Depth wise separable layer (kernel size = 7)
- Maxpool2D layer (poolsize =3, stride=2)
- Convolution layer
- 'Add layer' due to a residual connection. Infer connection points of the skip connection from the model summary and model plot.
- *Infer unspecified arguments from the summary** iii). The block defined in (ii) repeats 4 times. Note that in each repitition, the number of filters changes. Infer this from the model plot/summary.

iv). The last two layers are GlobalAveragePooling and Dense layers. The dense layer is the output layer (infer the number of neurons and the activation function).

Compile model with rmsprop as an optimizer with appropriate loss and metric for this respective problem. (1)

Fit the model with a batch_size of 32 for 20 epochs. (Don't use EarlyStopping callback). Use the validation dataset from the data you downloaded. We have specified a small no. of epochs because training may take time. Try running colab on GPU by going to Edit > Notebook accelerator > Hardware Accelerator > GPU. (1)

Return the history as a DataFrame. Show loss and accuracy for training and validation through appropriate plots.(1)

Evaluate the Model on test dataset from the data you downloaded.(1)

Note:

If you are using any parameter values or arguments apart from the ones mentioned or the ones that you must infer, state explicitly where and why you are using them.

Also verify that the total no. of params of your model are the same as that mentioned in model_summary txt file given to you

Double-click (or enter) to edit

Double-click (or enter) to edit

```
import random
random.seed(42)

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.utils import plot_model


!git clone -b master https://github.com/sumanthk94/DA_225-o.git
```

```
Cloning into 'DA_225-o'...
remote: Enumerating objects: 19066, done.
remote: Total 19066 (delta 0), reused 0 (delta 0), pack-reused 19066
```

```
Receiving objects: 100% (19066/19066), 570.44 MiB | 27.16 MiB/s, done.
Resolving deltas: 100% (3/3), done.
Updating files: 100% (19001/19001), done.
```

```
data_dir = 'DA_225-o/cats_vs_dogs_small'

train_path = data_dir + '/train'
validation_path = data_dir + '/validation'
test_path = data_dir + '/test'


# creating datasets using utility
from tensorflow.keras.utils import image_dataset_from_directory
train_dataset = image_dataset_from_directory(
               train_path,
                image_size=(180, 180), # Resize the images to (180,180)
                batch_size=32)
validation_dataset = image_dataset_from_directory(
                      validation_path,
                      image_size=(180, 180),
                      batch_size=32)
test_dataset = image_dataset_from_directory(
                test_path,
                image_size=(180, 180),
                batch_size=32)
```

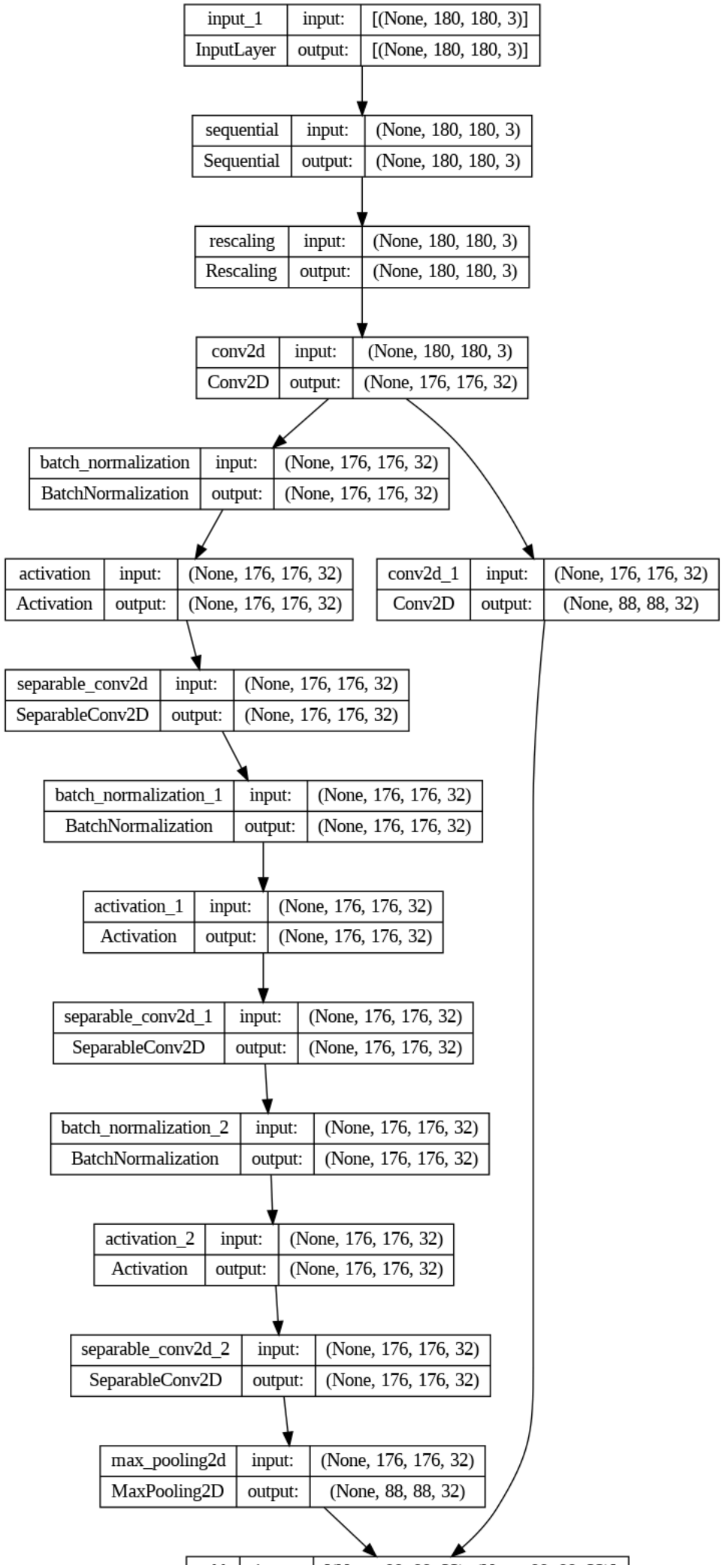
```
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 2000 files belonging to 2 classes.
```

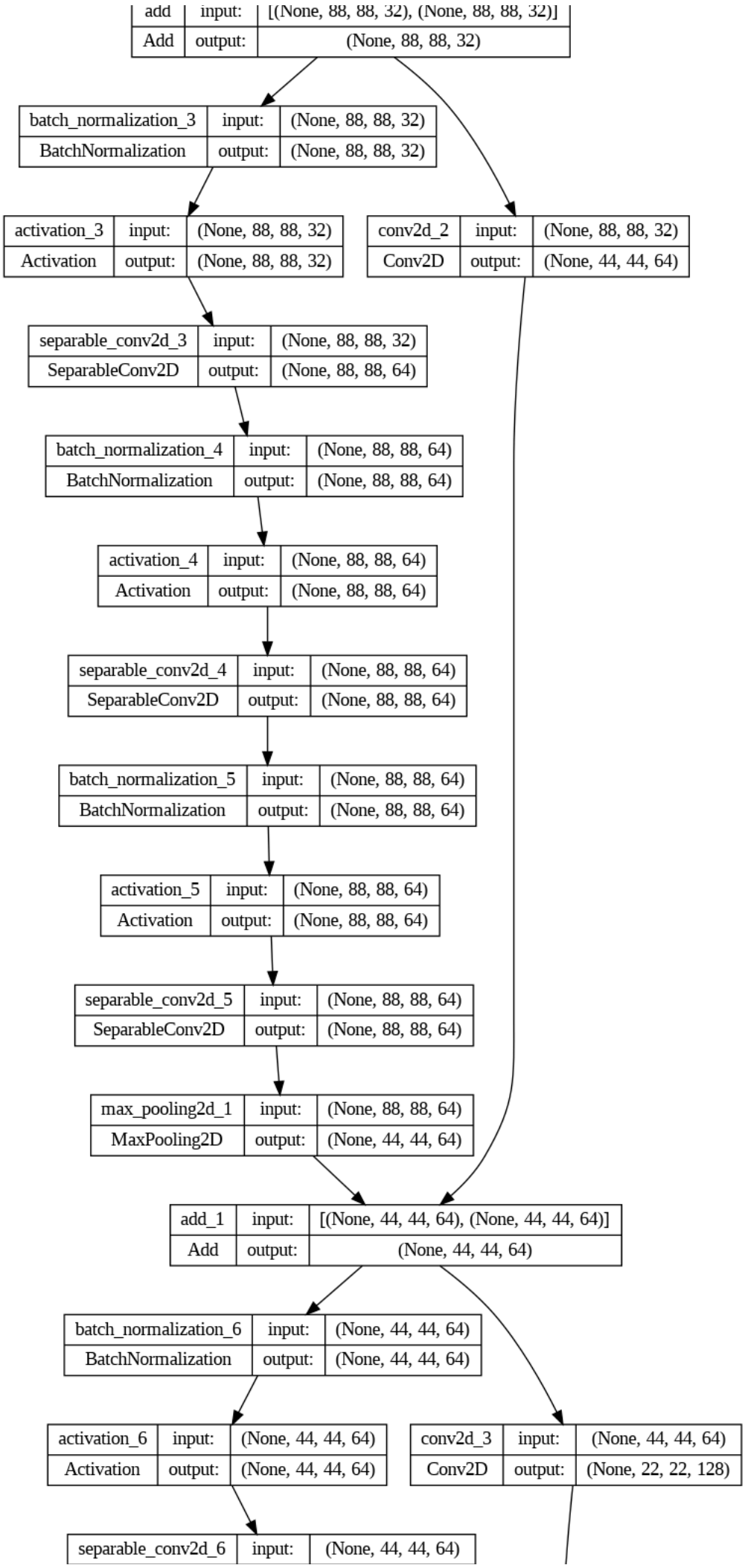
```
inputs = keras.Input(shape=(180, 180, 3))

flip="horizontal"
rotation=0.1
zoom=0.2
data_augmentation = keras.Sequential([
      layers.RandomFlip(flip),
      layers.RandomRotation(rotation),
      layers.RandomZoom(zoom)])
processing_layers = data_augmentation(inputs)
processing_layers = layers.Rescaling(1./255)(processing_layers)
processing_layers = layers.Conv2D(filters=32, kernel_size=5, use_bias=False)(processing_layers)



for size in [32, 64, 128, 256]:               # Repeated block. Very common practice
    residual = processing_layers

    processing_layers = layers.BatchNormalization()(processing_layers)             # We can also apply BN just before the activation
    processing_layers = layers.Activation("relu")(processing_layers)
    processing_layers = layers.SeparableConv2D(size, 3, padding="same", use_bias=False)(processing_layers)

    processing_layers = layers.BatchNormalization()(processing_layers)
    processing_layers = layers.Activation("relu")(processing_layers)
    processing_layers = layers.SeparableConv2D(size, 5, padding="same", use_bias=False)(processing_layers)

    processing_layers = layers.BatchNormalization()(processing_layers)
    processing_layers = layers.Activation("relu")(processing_layers)
    processing_layers = layers.SeparableConv2D(size, 7, padding="same", use_bias=False)(processing_layers)

    processing_layers = layers.MaxPooling2D(pool_size=3, strides = 2, padding="same")(processing_layers)

    residual = layers.Conv2D(size, 1, strides=2, padding="same", use_bias=False)(residual)

    processing_layers = layers.add([processing_layers, residual])

processing_layers = layers.GlobalAveragePooling2D()(processing_layers)
outputs = layers.Dense(1, activation="sigmoid")(processing_layers)
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
```

```
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
```

```
plot_model(model, show_shapes=True)
```


input_1 | input: | [(None, 180, 180, 3)]
InputLayer | output: | [(None, 180, 180, 3)]
sequential | input: | (None, 180, 180, 3)
Sequential | output: | (None, 180, 180, 3)
rescaling | input: | (None, 180, 180, 3)
Rescaling | output: | (None, 180, 180, 3)
conv2d | input: | (None, 180, 180, 3)
Conv2D | output: | (None, 176, 176, 32)
batch_normalization | input: | (None, 176, 176, 32)
BatchNormalization | output: | (None, 176, 176, 32)
activation | input: | (None, 176, 176, 32)
Activation | output: | (None, 176, 176, 32)
conv2d_1 | input: | (None, 176, 176, 32)
Conv2D | output: | (None, 88, 88, 32)
separable_conv2d | input: | (None, 176, 176, 32)
SeparableConv2D | output: | (None, 176, 176, 32)
batch_normalization_1 | input: | (None, 176, 176, 32)
BatchNormalization | output: | (None, 176, 176, 32)
activation_1 | input: | (None, 176, 176, 32)
Activation | output: | (None, 176, 176, 32)
separable_conv2d_1 | input: | (None, 176, 176, 32)
SeparableConv2D | output: | (None, 176, 176, 32)
batch_normalization_2 | input: | (None, 176, 176, 32)
BatchNormalization | output: | (None, 176, 176, 32)
activation_2 | input: | (None, 176, 176, 32)
Activation | output: | (None, 176, 176, 32)
separable_conv2d_2 | input: | (None, 176, 176, 32)
SeparableConv2D | output: | (None, 176, 176, 32)
max_pooling2d | input: | (None, 176, 176, 32)
MaxPooling2D | output: | (None, 88, 88, 32)


add | input: | [(None, 88, 88, 32), (None, 88, 88, 32)]
Add | output: | (None, 88, 88, 32)
batch_normalization_3 | input: | (None, 88, 88, 32)
BatchNormalization | output: | (None, 88, 88, 32)
activation_3 | input: | (None, 88, 88, 32)
Activation | output: | (None, 88, 88, 32)
conv2d_2 | input: | (None, 88, 88, 32)
Conv2D | output: | (None, 44, 44, 64)
separable_conv2d_3 | input: | (None, 88, 88, 32)
SeparableConv2D | output: | (None, 88, 88, 64)
batch_normalization_4 | input: | (None, 88, 88, 64)
BatchNormalization | output: | (None, 88, 88, 64)
activation_4 | input: | (None, 88, 88, 64)
Activation | output: | (None, 88, 88, 64)
separable_conv2d_4 | input: | (None, 88, 88, 64)
SeparableConv2D | output: | (None, 88, 88, 64)
batch_normalization_5 | input: | (None, 88, 88, 64)
BatchNormalization | output: | (None, 88, 88, 64)
activation_5 | input: | (None, 88, 88, 64)
Activation | output: | (None, 88, 88, 64)
separable_conv2d_5 | input: | (None, 88, 88, 64)
SeparableConv2D | output: | (None, 88, 88, 64)
max_pooling2d_1 | input: | (None, 88, 88, 64)
MaxPooling2D | output: | (None, 44, 44, 64)
add_1 | input: | [(None, 44, 44, 64), (None, 44, 44, 64)]
Add | output: | (None, 44, 44, 64)
batch_normalization_6 | input: | (None, 44, 44, 64)
BatchNormalization | output: | (None, 44, 44, 64)
activation_6 | input: | (None, 44, 44, 64)
Activation | output: | (None, 44, 44, 64)
conv2d_3 | input: | (None, 44, 44, 64)
Conv2D | output: | (None, 22, 22, 128)
separable_conv2d_6 | input: | (None, 44, 44, 64)