```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, Batc
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import binary_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, Sequential

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from PIL import Image
from IPython.core.display import display, HTML

# stop annoying tensorflow warning messages

import logging
logging.getLogger("tensorflow").setLevel(logging.ERROR)


from pathlib import PosixPath
import random
import matplotlib.pyplot as plt
%matplotlib inline

from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau


root_path = '/content/drive/MyDrive/archive/training_set/training_set'
data_dir = PosixPath(root_path)


image_count = len(list(data_dir.glob("*/*")))
print(f"Image count: {image_count}")
```

```
Image count: 8007
```

```
class_names = os.listdir(root_path)
class_names
```

```
['cats', 'dogs']
```

```
class_distribution = [len(os.listdir(root_path + '/' + name)) for name in class_names]
class_distribution
```

```
[4001, 4006]
```

```
seed = random.randint(0,100)
```

```
random.seed(seed)
seed = random.randint(0,1250)
print(f"Current seed : {seed}")
```

```
Current seed : 549
```

```
img_path=r'/content/drive/MyDrive/archive/training_set/training_set/cats/cat.1.jpg'
img=plt.imread(img_path)
print (img.shape)
plt.axis('off')
plt.imshow(img)
plt.show()
```

```
(280, 300, 3)
```

```
def preprocess (sdir, trsplit, random_seed):
    filepaths=[]
    labels=[]

    classlist=os.listdir(sdir)

    for klass in classlist:

        classpath=os.path.join(sdir,klass)
        flist=os.listdir(classpath)

        for f in flist:

            fpath=os.path.join(classpath,f)
            filepaths.append(fpath)
            labels.append(klass)

    Fseries=pd.Series(filepaths, name='filepaths')
    Lseries=pd.Series(labels, name='labels')
    df=pd.concat([Fseries, Lseries], axis=1)

    # split df into train_df and test_df
    dsplit=(1-trsplit)
    strat=df['labels']
    train_df, valid_df=train_test_split(df, train_size=trsplit, shuffle=True, random_state
```

```
    print('train_df length: ', len(train_df), '  valid_df length: ', len(valid_df))
    return train df. valid df
sdir=r'/content/drive/MyDrive/archive/training_set/training_set'
train_df, valid_df = preprocess(sdir, .8, seed)
```

```
 train_df length:  6405   valid_df length:  1602
```

```
filepaths=[]
labels=[]

classlist = os.listdir('/content/drive/MyDrive/archive/test_set/test_set')

for klass in classlist:

    classpath=os.path.join(sdir,klass)
    flist=os.listdir(classpath)

    for f in flist:

        fpath=os.path.join(classpath,f)
        filepaths.append(fpath)
        labels.append(klass)

Fseries=pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
test_df=pd.concat([Fseries, Lseries], axis=1)
print('test_df length: ', len(test_df))
```

```
 test_df length:  8007
```

```
img_size=(100,100)
channels=3
batch_size=32
img_shape=(img_size[0], img_size[1], channels)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and ler
test_steps=int(length/test_batch_size)
print ( 'test batch size: ' ,test_batch_size, '  test steps: ', test_steps)
```

```
 test batch size:  51   test steps:  157
```

```
def scalar(img):
    return img  # EfficientNet expects pixels in range 0 to 255 so no scaling is required

trgen=ImageDataGenerator(preprocessing_function=scalar, horizontal_flip=True)
tvgen=ImageDataGenerator(preprocessing_function=scalar)

train_gen=trgen.flow_from_dataframe( train_df, x_col='filepaths', y_col='labels', target_s
                                    color_mode='rgb', shuffle=True, batch_size=batch_size)

test_gen=tvgen.flow_from_dataframe( test_df, x_col='filepaths', y_col='labels', target_siz
                                    color_mode='rgb', shuffle=False, batch_size=test_batch
```

```
valid_gen=tvgen.flow_from_dataframe( valid_df, x_col='filepaths', y_col='labels', target_s
                                    color_mode='rgb', shuffle=True, batch_size=batch_size)
classes=list(train_gen.class_indices.keys())
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/dataframe_iterator.
  .format(n_invalid, x_col)
Found 6403 validated image filenames belonging to 2 classes.
Found 8005 validated image filenames belonging to 2 classes.
Found 1602 validated image filenames belonging to 2 classes.
```

```
class_count=len(classes)
print(class_count)
```

```
2
```

```
model_name='EfficientNetB3'
base_model=tf.keras.applications.EfficientNetB3(include_top=False, weights="imagenet",inpu
x=base_model.output
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(256, kernel_regularizer = regularizers.l2(l = 0.016),activity_regularizer=regula
                bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.45, seed=123)(x)
x = Flatten()(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)
model.compile(Adamax(learning_rate=.001), loss='categorical_crossentropy', metrics=['accur
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnet
43941888/43941136 [==============================] - 0s 0us/step
43950080/43941136 [==============================] - 0s 0us/step
```

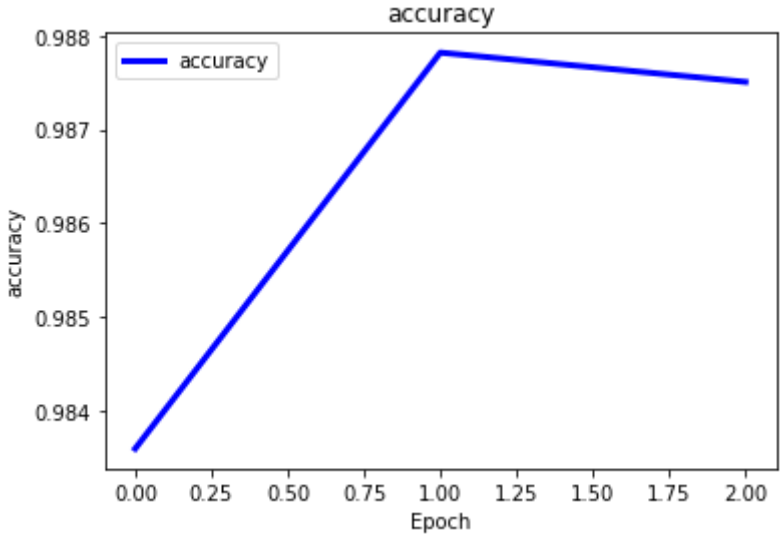
```
early_stopping_monitor = EarlyStopping(monitor='val_accuracy', patience=5, restore_best_we

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                            patience=5,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)

best_model = ModelCheckpoint('/content/drive/MyDrive/bestmodel.hdf5', monitor='accuracy',
best_val_acc = ModelCheckpoint('/content/drive/MyDrive/best_val_acc.hdf5', monitor='val_ac


epochs = 3
history=model.fit(x=train_gen,  epochs=epochs, verbose=1, batch_size = 256,  validation_da
    best_model, best_val_acc],
    shuffle=True,  initial_epoch=0)
```

```
Epoch 1/3
201/201 [==============================] - 902s 4s/step - loss: 0.5074 - accuracy: 0
Epoch 2/3
201/201 [==============================] - 934s 5s/step - loss: 0.3468 - accuracy: 0
```

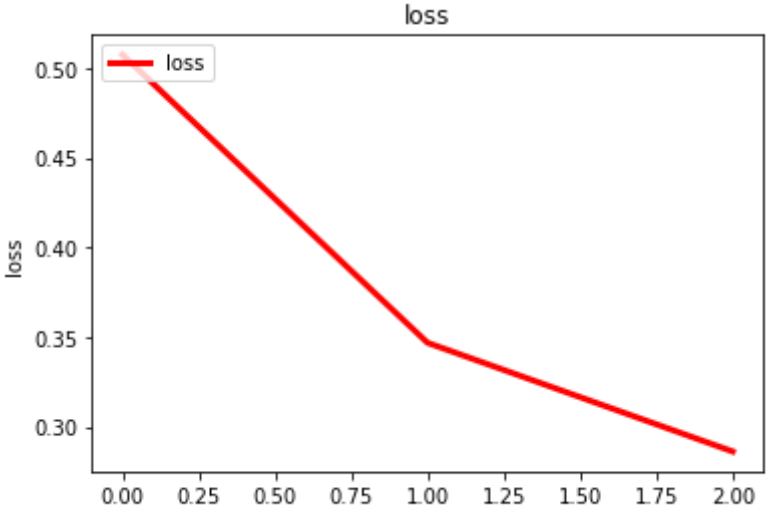
```
Epoch 3/3
201/201 [==============================] - 898s 4s/step - loss: 0.2863 - accuracy: 0
```

```
def visualization(name,h,color):
  t = h.history[name]
  my_max = max(t)
  my_min = min(t)
  print(f'Name : {name} max : {my_max} min : {my_min}')
  plt.plot(t,color=color,linewidth=3.0)
  plt.title(name)
  plt.ylabel(name)
  plt.xlabel('Epoch')
  plt.legend([name],loc='upper left')
  plt.show()


visualization('accuracy',history,'Blue')
visualization('loss',history,'Red')
visualization('val_accuracy',history,'Green')
visualization('val_loss',history,'Black')
```

```
Name : accuracy max : 0.98781818151474 min : 0.983601450920105
```




```
Name : loss max : 0.5073956847190857 min : 0.28625068068504333
```




```
import seaborn as sns
from sklearn.metrics import f1_score, confusion_matrix

model.load_weights('/content/drive/MyDrive/bestmodel.hdf5')
accuracy=model.evaluate (test_gen, steps=test_steps)[1]
print(f'accuracy on test set = {accuracy:6.2f}')
```

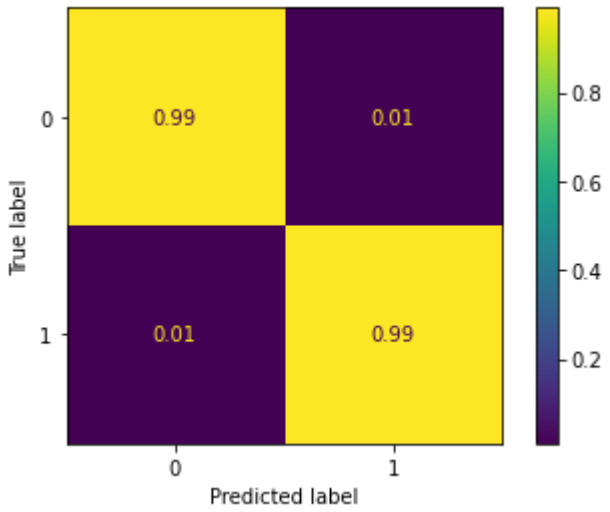
```
157/157 [==============================] - 216s 1s/step - loss: 0.2902 - accuracy: 0
accuracy on test set =   0.99
```

```
from sklearn.metrics import classification_report

predictions = model.predict_generator(test_gen, steps=test_steps)
test_steps_per_epoch = np.math.ceil(test_gen.samples / test_batch_size)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_gen.classes
class_labels = list(test_gen.class_indices.keys())
report = classification_report(true_classes, predicted_classes, target_names=class_labels)
print(report)
```

```
cm=confusion_matrix(true_classes,predicted_classes)
print(cm)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.
  """Entry point for launching an IPython kernel.
              precision    recall  f1-score   support

        cats       0.99      0.99      0.99      4000
        dogs       0.99      0.99      0.99      4005

    accuracy                           0.99      8005
   macro avg       0.99      0.99      0.99      8005
weighted avg       0.99      0.99      0.99      8005

[[3976   24]
 [  44 3961]]
```

```
from sklearn.metrics import ConfusionMatrixDisplay
cm_norm = np.round(cm/np.sum(cm,axis=1).reshape(-1,1),2)
cm_display = ConfusionMatrixDisplay(cm_norm)
cm_display.plot()
plt.show()
```




```
sns.heatmap(cm_norm,cmap='Greens',annot=True,
            cbar_kws={'orientation' : 'vertical','label' : 'Color bar'},
            fmt='.2f'
            )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


```
model.load_weights('/content/drive/MyDrive/best_val_acc.hdf5')
accuracy=model.evaluate (test_gen, steps=test_steps)[1]
print(f'accuracy on test set = {accuracy:6.2f}')
```

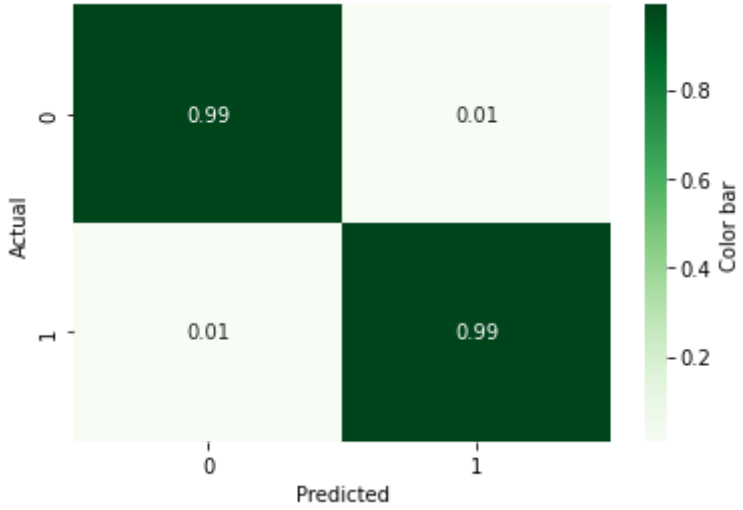
```
157/157 [==============================] - 227s 1s/step - loss: 0.2902 - accuracy: 0
accuracy on test set =   0.99
```

```
predictions = model.predict_generator(test_gen, steps=test_steps)
test_steps_per_epoch = np.math.ceil(test_gen.samples / test_batch_size)
predicted_classes = np.argmax(predictions, axis=1)
true_classes = test_gen.classes
class_labels = list(test_gen.class_indices.keys())
report = classification_report(true_classes, predicted_classes, target_names=class_labels)
print(report)

cm=confusion_matrix(true_classes,predicted_classes)
print(cm)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.
  """Entry point for launching an IPython kernel.
              precision    recall  f1-score   support

        cats       0.99      0.99      0.99      4000
        dogs       0.99      0.99      0.99      4005

    accuracy                           0.99      8005
   macro avg       0.99      0.99      0.99      8005
weighted avg       0.99      0.99      0.99      8005

[[3976   24]
 [  44 3961]]
```

```
cm_norm = np.round(cm/np.sum(cm,axis=1).reshape(-1,1),2)
cm_display = ConfusionMatrixDisplay(cm_norm)
cm_display.plot()
plt.show()
```


```
sns.heatmap(cm_norm,cmap='Greens',annot=True,
            cbar_kws={'orientation' : 'vertical','label' : 'Color bar'},
            fmt='.2f'
            )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



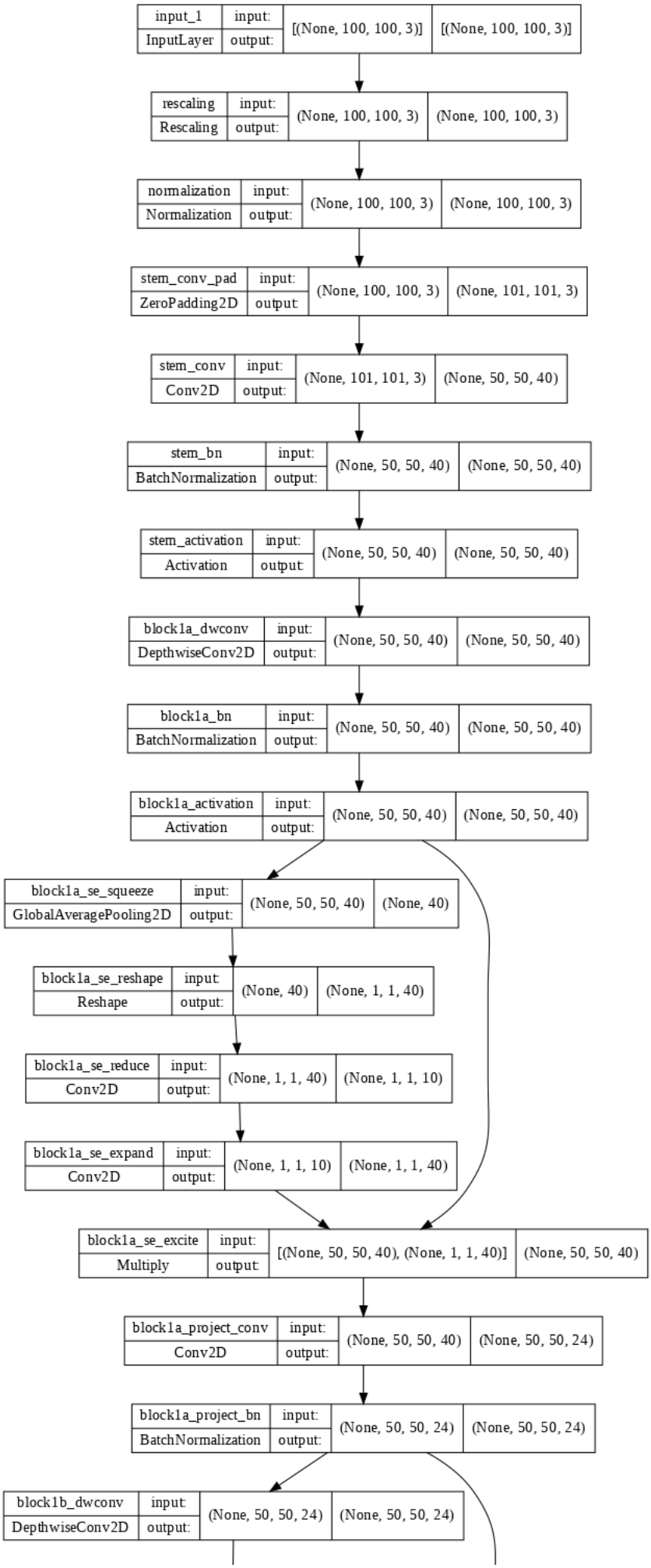

```
model.summary()
```

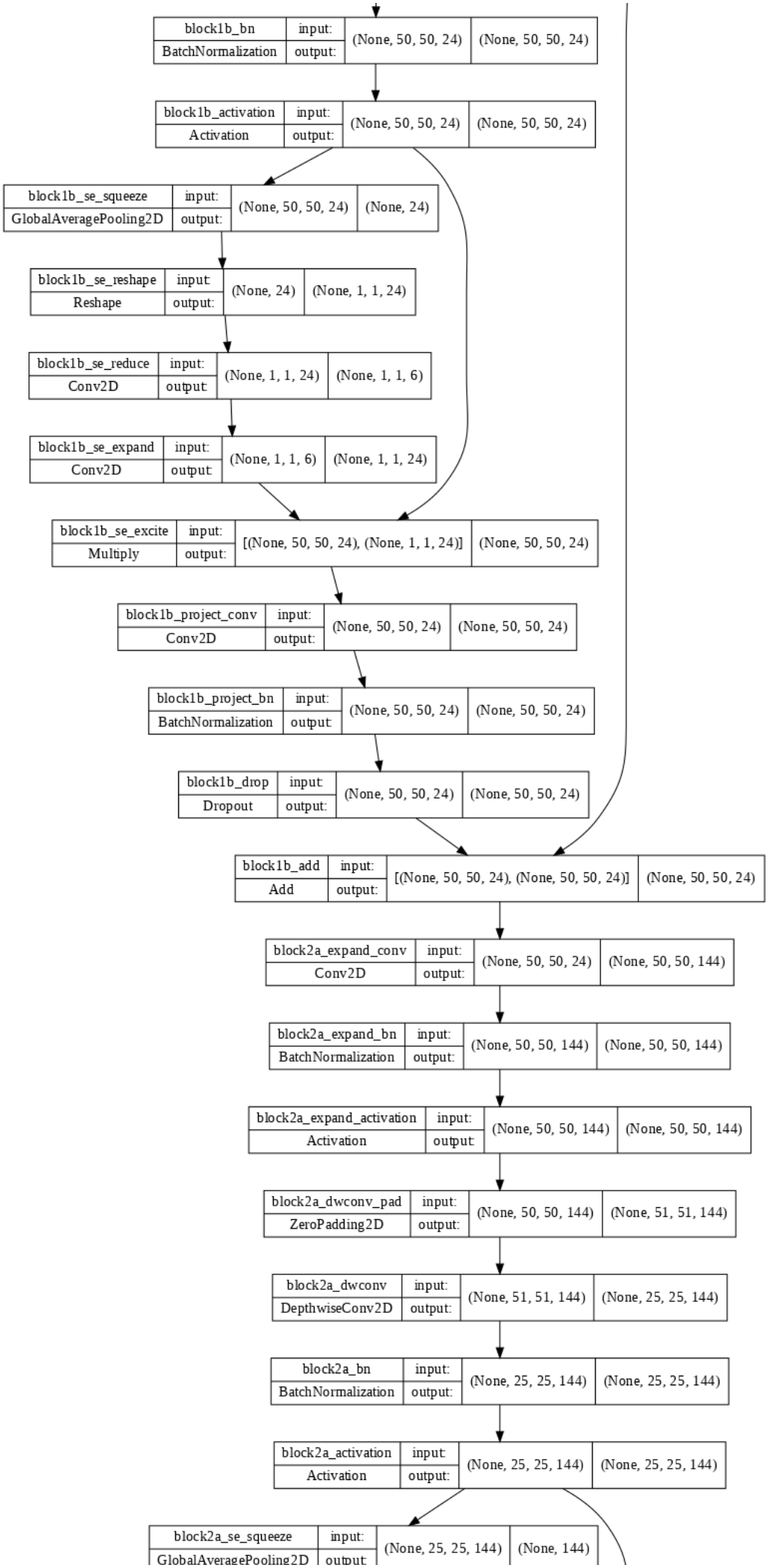
```
ivation)

block7b_dwconv (DepthwiseConv2  (None, 4, 4, 2304)  20736       ['block7b_expand_
D)                                                              ']

block7b_bn (BatchNormalization  (None, 4, 4, 2304)  9216        ['block7b_dwconv
)

block7b_activation (Activation  (None, 4, 4, 2304)  0           ['block7b_bn[0][
)

block7b_se_squeeze (GlobalAver  (None, 2304)        0           ['block7b_activa
agePooling2D)

block7b_se_reshape (Reshape)   (None, 1, 1, 2304)   0           ['block7b_se_squ

block7b_se_reduce (Conv2D)     (None, 1, 1, 96)     221280      ['block7b_se_res

block7b_se_expand (Conv2D)     (None, 1, 1, 2304)   223488      ['block7b_se_red

block7b_se_excite (Multiply)   (None, 4, 4, 2304)   0           ['block7b_activa
                                                                 'block7b_se_exp

block7b_project_conv (Conv2D)  (None, 4, 4, 384)    884736      ['block7b_se_exc

block7b_project_bn (BatchNorma  (None, 4, 4, 384)   1536        ['block7b_projec
```

```
block7b_project_bn (BatchNorma  (None, 4, 4, 384)   1536        ['block7b_projec
lization)

block7b_drop (Dropout)          (None, 4, 4, 384)   0           ['block7b_projec

block7b_add (Add)               (None, 4, 4, 384)   0           ['block7b_drop[0
                                                                 'block7a_projec

top_conv (Conv2D)               (None, 4, 4, 1536)  589824      ['block7b_add[0]

top_bn (BatchNormalization)     (None, 4, 4, 1536)  6144        ['top_conv[0][0]

top_activation (Activation)     (None, 4, 4, 1536)  0           ['top_bn[0][0]']

max_pool (GlobalMaxPooling2D)   (None, 1536)        0           ['top_activation

batch_normalization (BatchNorm  (None, 1536)        6144        ['max_pool[0][0]
alization)

dense (Dense)                   (None, 256)         393472      ['batch_normaliz

dropout (Dropout)               (None, 256)         0           ['dense[0][0]']

flatten (Flatten)               (None, 256)         0           ['dropout[0][0]'

dense_1 (Dense)                 (None, 2)           514         ['flatten[0][0]'

==================================================================================
Total params: 11,183,665
Trainable params: 11,093,290
Non-trainable params: 90,375
__________________________________________________________________________________
```

```
tf.keras.utils.plot_model(model,show_shapes=True)
```

```
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.75985 to fit
```




block1b_bn | BatchNormalization | input: | output: | (None, 50, 50, 24) | (None, 50, 50, 24)
block1b_activation | Activation | input: | output: | (None, 50, 50, 24) | (None, 50, 50, 24)
block1b_se_squeeze | GlobalAveragePooling2D | input: | output: | (None, 50, 50, 24) | (None, 24)
block1b_se_reshape | Reshape | input: | output: | (None, 24) | (None, 1, 1, 24)
block1b_se_reduce | Conv2D | input: | output: | (None, 1, 1, 24) | (None, 1, 1, 6)
block1b_se_expand | Conv2D | input: | output: | (None, 1, 1, 6) | (None, 1, 1, 24)
block1b_se_excite | Multiply | input: | output: | [(None, 50, 50, 24), (None, 1, 1, 24)] | (None, 50, 50, 24)
block1b_project_conv | Conv2D | input: | output: | (None, 50, 50, 24) | (None, 50, 50, 24)
block1b_project_bn | BatchNormalization | input: | output: | (None, 50, 50, 24) | (None, 50, 50, 24)
block1b_drop | Dropout | input: | output: | (None, 50, 50, 24) | (None, 50, 50, 24)
block1b_add | Add | input: | output: | [(None, 50, 50, 24), (None, 50, 50, 24)] | (None, 50, 50, 24)
block2a_expand_conv | Conv2D | input: | output: | (None, 50, 50, 24) | (None, 50, 50, 144)
block2a_expand_bn | BatchNormalization | input: | output: | (None, 50, 50, 144) | (None, 50, 50, 144)
block2a_expand_activation | Activation | input: | output: | (None, 50, 50, 144) | (None, 50, 50, 144)
block2a_dwconv_pad | ZeroPadding2D | input: | output: | (None, 50, 50, 144) | (None, 51, 51, 144)
block2a_dwconv | DepthwiseConv2D | input: | output: | (None, 51, 51, 144) | (None, 25, 25, 144)
block2a_bn | BatchNormalization | input: | output: | (None, 25, 25, 144) | (None, 25, 25, 144)
block2a_activation | Activation | input: | output: | (None, 25, 25, 144) | (None, 25, 25, 144)
block2a_se_squeeze | GlobalAveragePooling2D | input: | output: | (None, 25, 25, 144) | (None, 144)