1. Download all the data in this folder https://drive.google.com/open?id=1Z4TyI7FcFVEx8qdl
path/to/the/image.tif,category

where the categories are numbered 0 to 15, in the following order:

0 letter
1 form
2 email
3 handwritten
4 advertisement
5 scientific report
6 scientific publication
7 specification
8 file folder
9 news article
10 budget
11 invoice
12 presentation
13 questionnaire
14 resume
15 memo

2. On this image data, you have to train 3 types of models as given below. You have to spl

3. Try not to load all the images into memory, use the gernarators that we have given the
or you can use this method also
https://medium.com/@vijayabhaskar96/tutorial-on-keras-imagedatagenerator-with-flow-from-da

https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c

4. You are free to choose Learning rate, optimizer, loss function, image augmentation, any

5. Use tensorboard for every model and analyse your gradients. (you need to upload the scr

Note: fit_genarator() method will have problems with the tensorboard histograms, try to del

6. You can check about Transfer Learning in this link - https://blog.keras.io/building-pow

```
#!wget --header="Host: doc-08-6g-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.
```

```
get_ipython().system_raw("unrar x rvl-cdip.rar")
```

```
%load_ext tensorboard
```

```
from keras_preprocessing.image import ImageDataGenerator
from tensorflow import keras
from sklearn.model_selection import train_test_split
import pandas as pd
```

```
df = pd.read_csv('/content/labels_final.csv')

train_data, valid_data = train_test_split(df, test_size=0.3, random_state=42)

#data augmentation
datagen = ImageDataGenerator(rescale=1./255)

#get images from different directories
train_generator = datagen.flow_from_dataframe(dataframe = train_data,
                  directory="data_final", x_col="path", y_col="label",
                  class_mode="raw", target_size=(256,256), batch_size=32)

valid_generator = datagen.flow_from_dataframe(dataframe = valid_data,
                  directory="data_final", x_col="path", y_col="label",
                  class_mode="raw", target_size=(256,256), batch_size=32)
```

```
Found 33600 validated image filenames.
Found 14400 validated image filenames.
```

## Model-1

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weigl
2. After VGG-16 network without FC layers, add a new Conv block ( 1 Conv layer and 1 Maxpoo
3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> M**
4. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

```
#Model 1
import datetime
from keras.models import Model

vgg_model = keras.applications.VGG16(weights='imagenet',
                                     include_top=False,
```

```
                                input_shape=(256, 256, 3))

x = vgg_model.output
x = keras.layers.Conv2D(32, (3, 3),padding  ='same')(x)
x = keras.layers.MaxPooling2D(pool_size=(2, 2))(x)
x = keras.layers.Flatten()(x)
x = keras.layers.Dense(128, activation='relu')(x)
x = keras.layers.Dropout(.5)(x)
x = keras.layers.Dense(64, activation='relu')(x)
x = keras.layers.Dropout(.5)(x)
preds = keras.layers.Dense(16, activation='softmax')(x)
model1 = Model(inputs = vgg_model.input,outputs=preds)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
58892288/58889256 [==============================] - 0s 0us/step
```

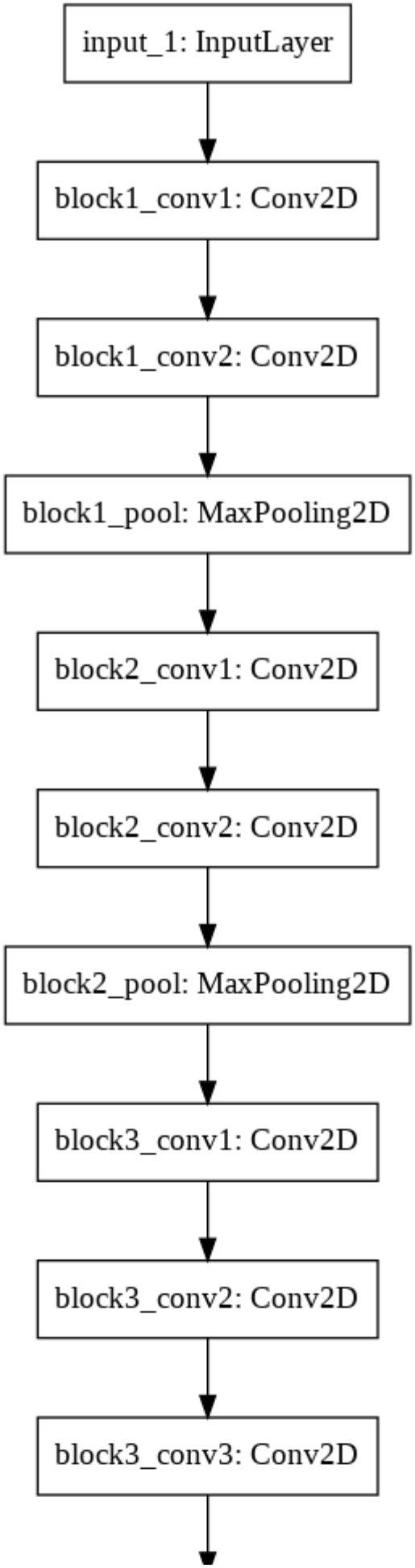
```
for layer in model1.layers[:19]:
    layer.trainable=False
for layer in model1.layers[19:]:
    layer.trainable=True

model1.summary()
```

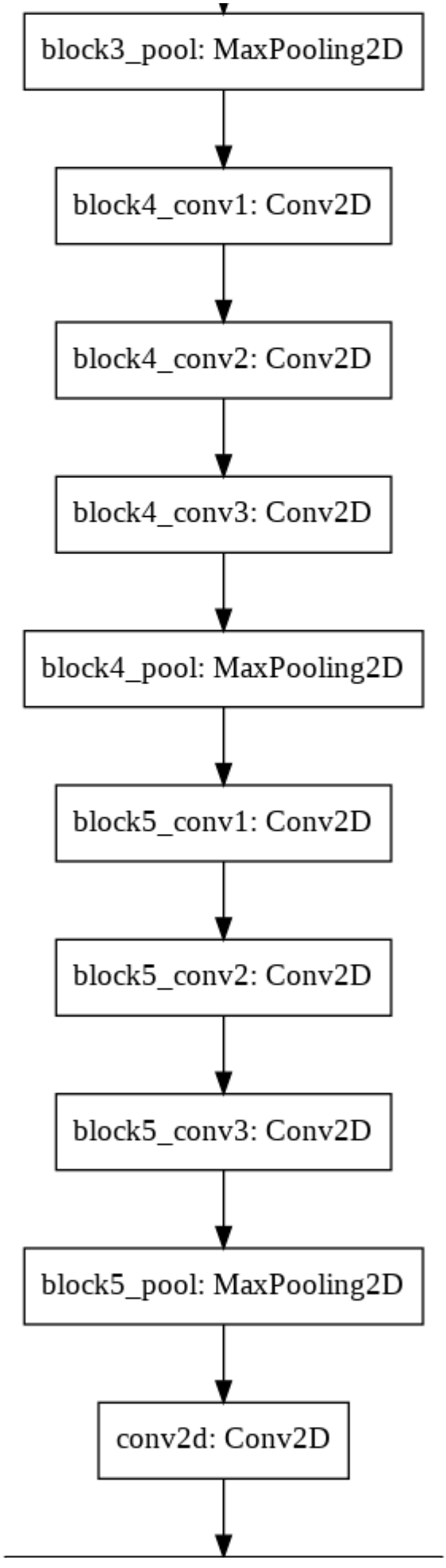
```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 256, 256, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 256, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 128, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 128, 128, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 128, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 64, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 64, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 64, 64, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 64, 64, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 32, 32, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 32, 32, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 32, 32, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 32, 32, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 16, 16, 512)       0         
```

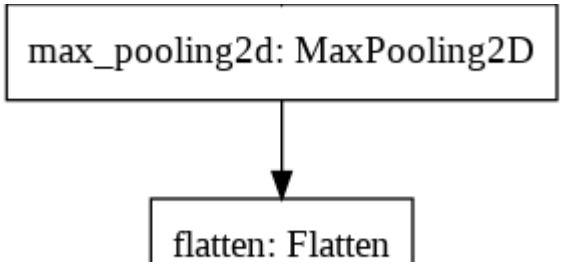
```
import tensorflow as tf
model1.compile(optimizer= 'adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])

#tensorflow callback
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_gr

#early stopping callback
early_stopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2)


STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size

with tf.device('/device:GPU:0'):
  model1.fit_generator(generator=train_generator,
                       steps_per_epoch=STEP_SIZE_TRAIN,
                       validation_data=valid_generator,
                       validation_steps=STEP_SIZE_VALID,
                       callbacks = [tensorboard_callback, early_stopping],
                       epochs=5)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard
WARNING:tensorflow:From <ipython-input-10-f5fb2a2567c6>:21: Model.fit_generator (from t
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/5
   1/1050 [..............................] - ETA: 0s - loss: 3.3809 - accuracy: 0.0312W
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
1050/1050 [==============================] - 541s 516ms/step - loss: 2.0371 - accuracy:
Epoch 2/5
1050/1050 [==============================] - 541s 515ms/step - loss: 1.5413 - accuracy:
Epoch 3/5
1050/1050 [==============================] - 540s 515ms/step - loss: 1.3827 - accuracy:
Epoch 4/5
1050/1050 [==============================] - 540s 515ms/step - loss: 1.2765 - accuracy:
Epoch 5/5
1050/1050 [==============================] - 541s 515ms/step - loss: 1.1900 - accuracy:
```

```
keras.utils.plot_model(
    model1, to_file='model1.png', show_shapes=False, show_layer_names=True,
    rankdir='TB', expand_nested=False, dpi=96
)
```


input_1: InputLayer
block1_conv1: Conv2D
block1_conv2: Conv2D
block1_pool: MaxPooling2D
block2_conv1: Conv2D
block2_conv2: Conv2D
block2_pool: MaxPooling2D
block3_conv1: Conv2D
block3_conv2: Conv2D
block3_conv3: Conv2D


block3_pool: MaxPooling2D
block4_conv1: Conv2D
block4_conv2: Conv2D
block4_conv3: Conv2D
block4_pool: MaxPooling2D
block5_conv1: Conv2D
block5_conv2: Conv2D
block5_conv3: Conv2D
block5_pool: MaxPooling2D
conv2d: Conv2D




```
#!rm -rf ./logs
```

```
%tensorboard --logdir /content/logs/fit/20200827-164553
```

```
for layer in model_2.layers[:13]:
    layer.trainable=False
for layer in model_2.layers[13:]:
    layer.trainable=True

model_2.summary()
```

- The dataset difference between the pretrained model and our models are also high regarding sizes, variance of categories.


## Observations:

- The validation accuracy for Mode1 1 = 0.69
- The imagenet dataset contains 14 million images and our modle trained on 33600 images, a huge differences in dataset
- Maybe the imagenet dataset has less category of file type data

## Model-2


1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weigl
2. After VGG-16 network without FC layers, don't use FC layers, use conv layers only as Fu
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identic
3. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the V

```
vgg_model = keras.applications.VGG16(weights='imagenet',
                                     include_top=False,
                                     input_shape=(256, 256, 3))

x2 = vgg_model.output
x2 = keras.layers.Conv2D(4096, (8, 8),padding  ='valid')(x2)
x2 = keras.layers.Conv2D(4096, (1, 1),padding  ='valid')(x2)
x2 = keras.layers.Flatten()(x2)
preds = keras.layers.Dense(16, activation='softmax')(x2)

model_2 = Model(inputs = vgg_model.input,outputs=preds)

for layer in model_2.layers[:19]:
    layer.trainable=False
for layer in model_2.layers[19:]:
    layer.trainable=True

model_2.summary()
```

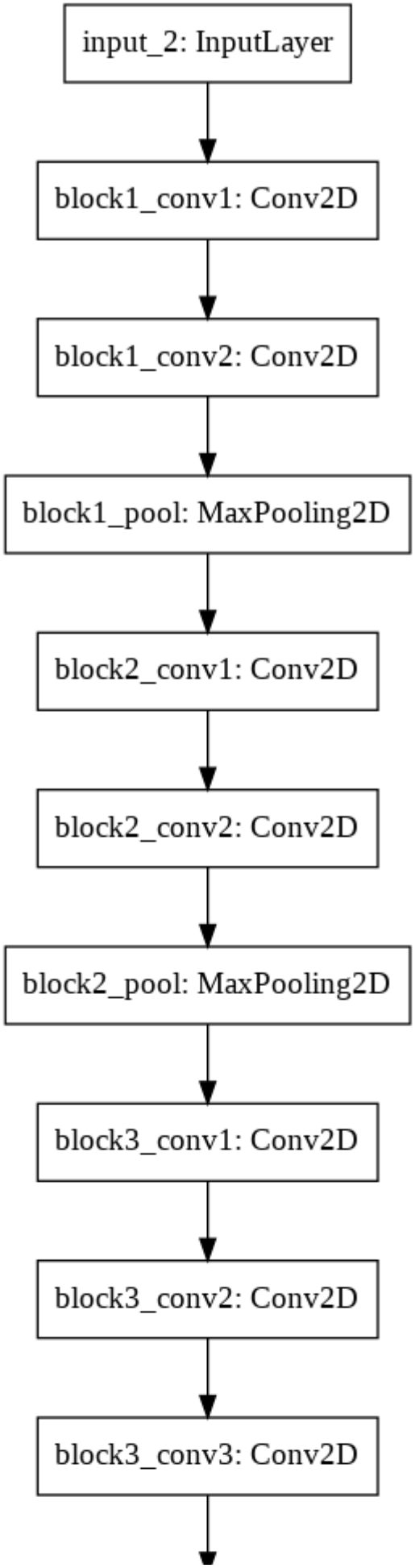
```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 256, 256, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 256, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 128, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 128, 128, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 128, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 64, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 64, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 64, 64, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 64, 64, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 32, 32, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 32, 32, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 16, 16, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 1, 1, 4096)        134221824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 1, 4096)        16781312
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
```

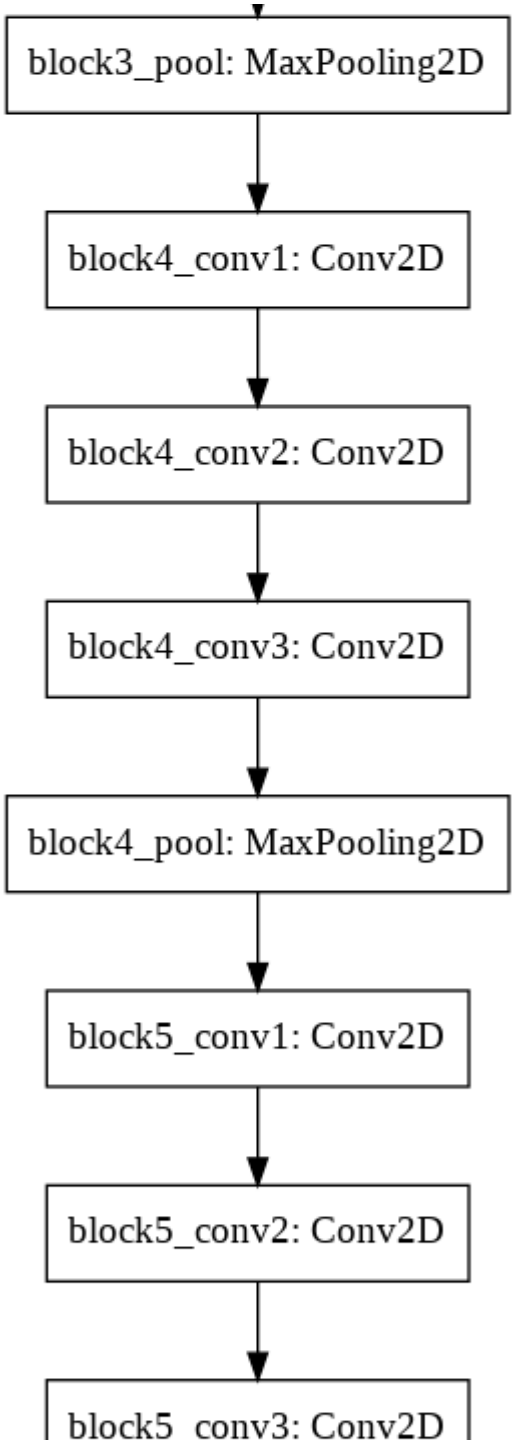
```
model_2.compile(optimizer= 'adam',loss='sparse_categorical_crossentropy',metrics=['accuracy']

#tensorflow callback
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_gr

#early stopping callback
early_stopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2)
```

```
early_stopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2)

model_2.fit_generator(generator=train_generator,
                      steps_per_epoch=STEP_SIZE_TRAIN,
                      validation_data=valid_generator,
                      validation_steps=STEP_SIZE_VALID,
                      callbacks = [tensorboard_callback, early_stopping],
                      epochs=5)
```

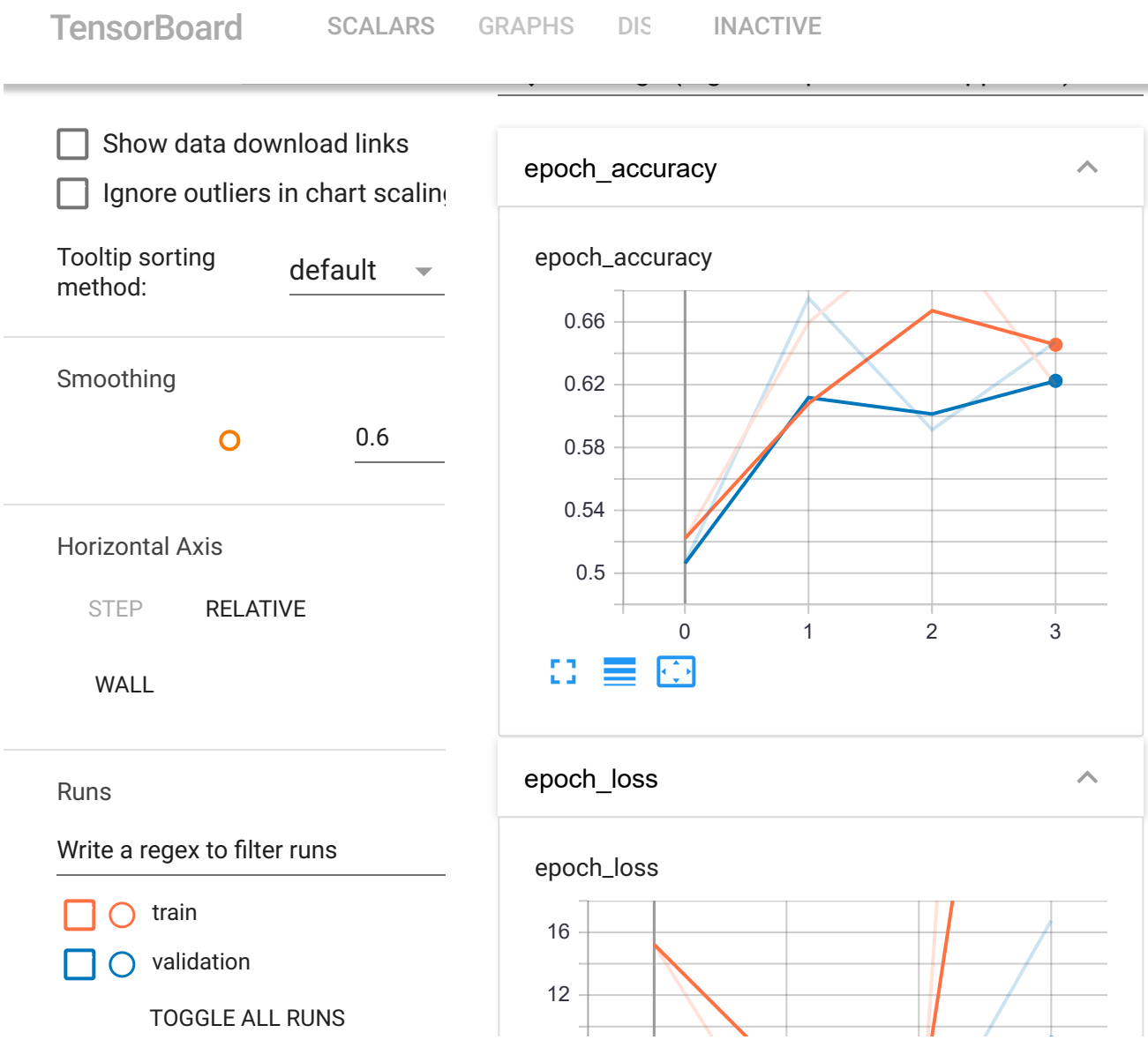
```
flow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.

==========================] - 1096s 1s/step - loss: 15.2254 - accuracy: 0.5221 - val_lo

==========================] - 1091s 1s/step - loss: 1.7481 - accuracy: 0.6596 - val_los

==========================] - 1093s 1s/step - loss: 0.9951 - accuracy: 0.7239 - val_los

==========================] - 1094s 1s/step - loss: 125.0922 - accuracy: 0.6199 - val_l
thon.keras.callbacks.History at 0x7f17ceb93588>
```

```
keras.utils.plot_model(
    model_2, to_file='model2.png', show_shapes=False, show_layer_names=True,
    rankdir='TB', expand_nested=False, dpi=96)
```


input_2: InputLayer
block1_conv1: Conv2D
block1_conv2: Conv2D
block1_pool: MaxPooling2D
block2_conv1: Conv2D
block2_conv2: Conv2D
block2_pool: MaxPooling2D
block3_conv1: Conv2D
block3_conv2: Conv2D
block3_conv3: Conv2D




```
%tensorboard --logdir logs/fit/20200827-173114
```




## Observations:

- In Model 2 the validation accuracy = 0.64
- Even though the only difference in vgg_16 and this model was last two layers in vgg-16 were dense and model 2 had convoluted data
- Maybe imagenet data doesn't consist variety of file type images

## Model-3

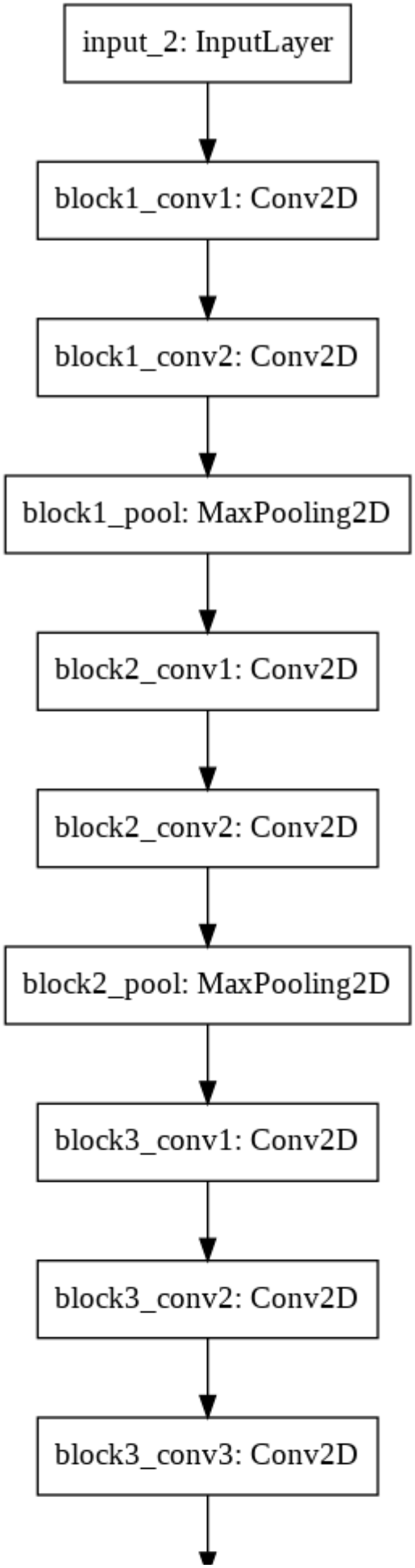
```
1. Use same network as Model-2 'INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers :
```

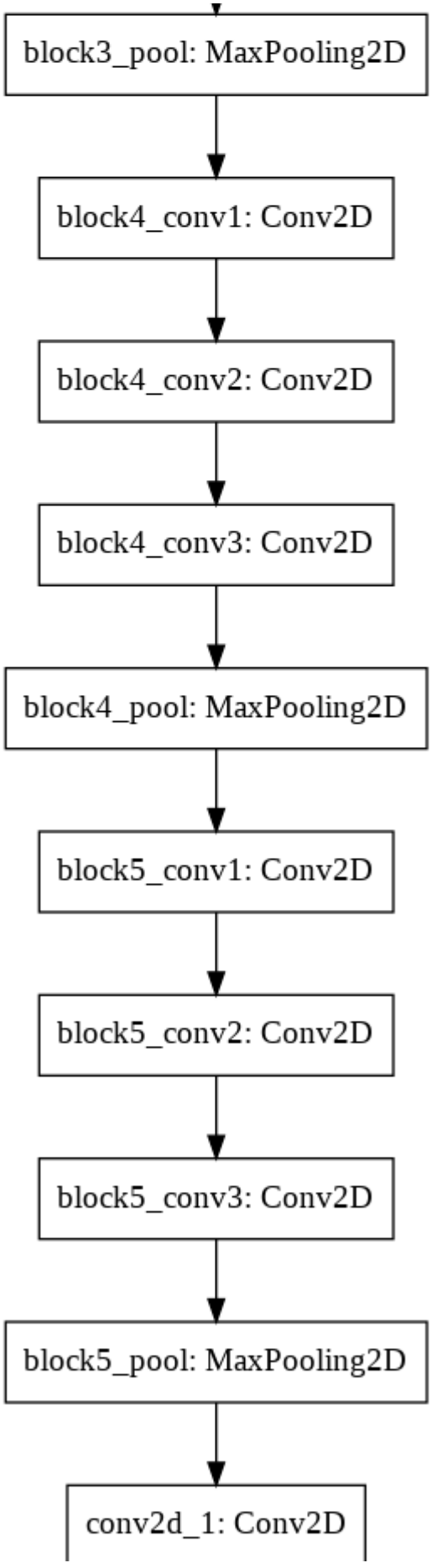
```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 256, 256, 3)]     0
```

```
model_2.compile(optimizer= 'adam',loss='sparse_categorical_crossentropy',metrics=['accuracy']

#tensorflow callback
log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_gr

#early stopping callback
early_stopping = keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2)

model_2.fit_generator(generator=train_generator,
                      steps_per_epoch=STEP_SIZE_TRAIN,
                      validation_data=valid_generator,
                      validation_steps=STEP_SIZE_VALID,
                      callbacks = [tensorboard_callback, early_stopping],
                      epochs=5)
```

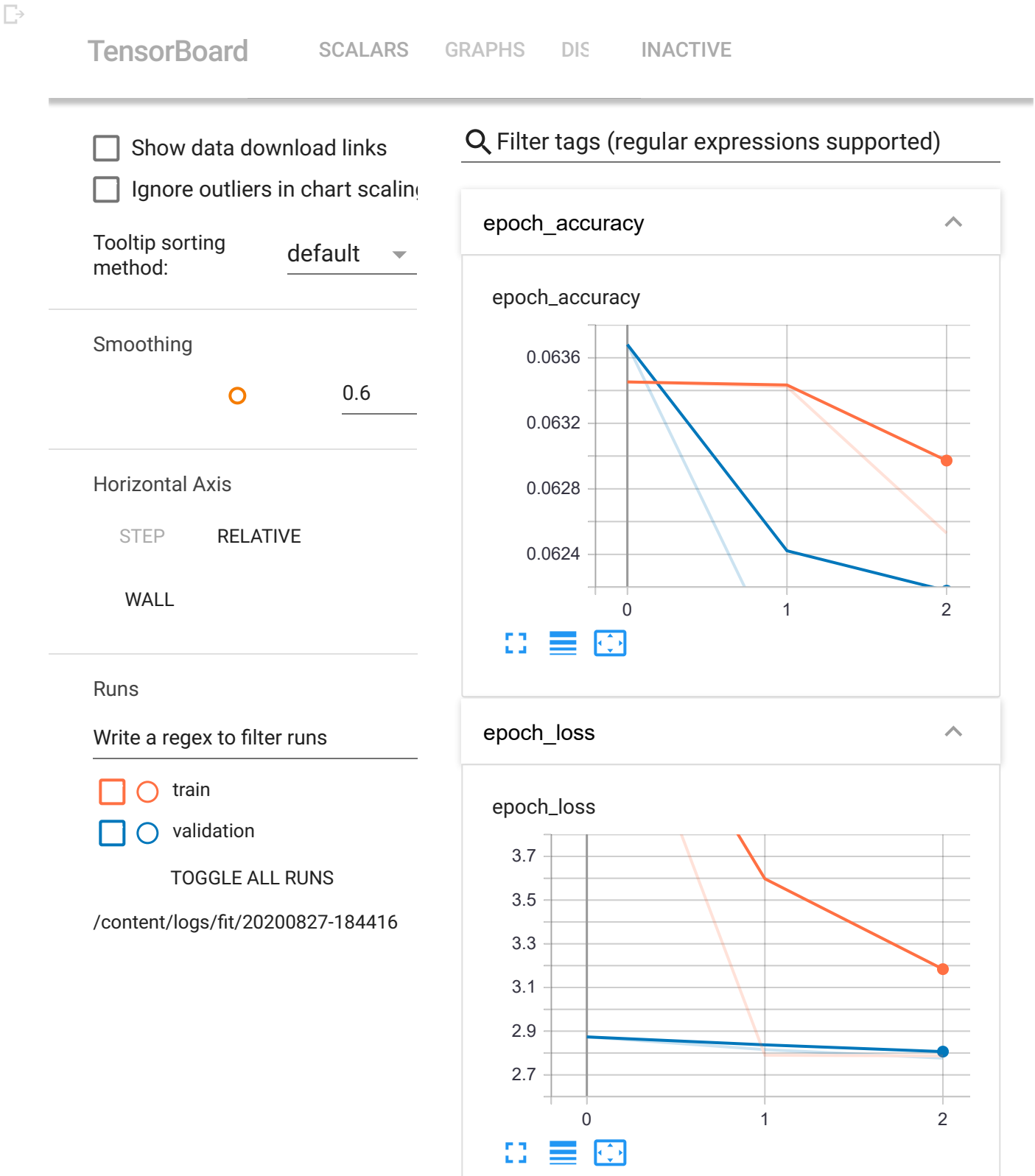
```
sorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.

==============================] - 1652s 2s/step - loss: 4.9436 - accuracy: 0.0635 - val_

==============================] - 1643s 2s/step - loss: 2.7900 - accuracy: 0.0634 - val_

==============================] - 1652s 2s/step - loss: 2.7869 - accuracy: 0.0625 - val_
.python.keras.callbacks.History at 0x7f17cc1229b0>
```

```
keras.utils.plot_model(
    model_2, to_file='model_2.png', show_shapes=False, show_layer_names=True,
    rankdir='TB', expand_nested=False, dpi=96)
```


input_2: InputLayer
block1_conv1: Conv2D
block1_conv2: Conv2D
block1_pool: MaxPooling2D
block2_conv1: Conv2D
block2_conv2: Conv2D
block2_pool: MaxPooling2D
block3_conv1: Conv2D
block3_conv2: Conv2D
block3_conv3: Conv2D




```
%tensorboard --logdir /content/logs/fit/20200827-184416
```




## Observations:

- The validation accuracy for model 3 is 0.0619
- In these model we have also trained the pre trained layers of vgg-16 models
- The accuracy is so low than other two models maybe the weights generated by pretrained models were not useful for the images we have.