1. Download all the data in this folder https://drive.google.com/open?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu. it contains two file bo
   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 split the data into Train and Validation dat

3. Try not to load all the images into memory, use the gernarators that we have given the reference notebooks to load the batch of
or you can use this method also
https://medium.com/@vijayabhaskar96/tutorial-on-keras-imagedatagenerator-with-flow-from-dataframe-8bd5776e45c1

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 hyperparameters. but you have to use the

5. Use tensorboard for every model and analyse your gradients. (you need to upload the screenshots for each model for evaluation)

Note: fit_genarator() method will have problems with the tensorboard histograms, try to debug it, if you could not do use histgrams

6. You can check about Transfer Learning in this link - https://blog.keras.io/building-powerful-image-classification-models-using-v

Model-1

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weights.
2. After VGG-16 network without FC layers, add a new Conv block (1 Conv layer and 1 Maxpooling), 2 FC layers and a output layer t

3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool Layer --> 2 FC layers --> Output L**
4. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

Model-2

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with Imagenet trained weights.
2. After VGG-16 network without FC layers, don't use FC layers, use conv layers only as Fully connected layer. any FC layer can be
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC layers, 1 output layer for 16 cl
3. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

Model-3

1. Use same network as Model-2 '**INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer**' and tra

Model 1

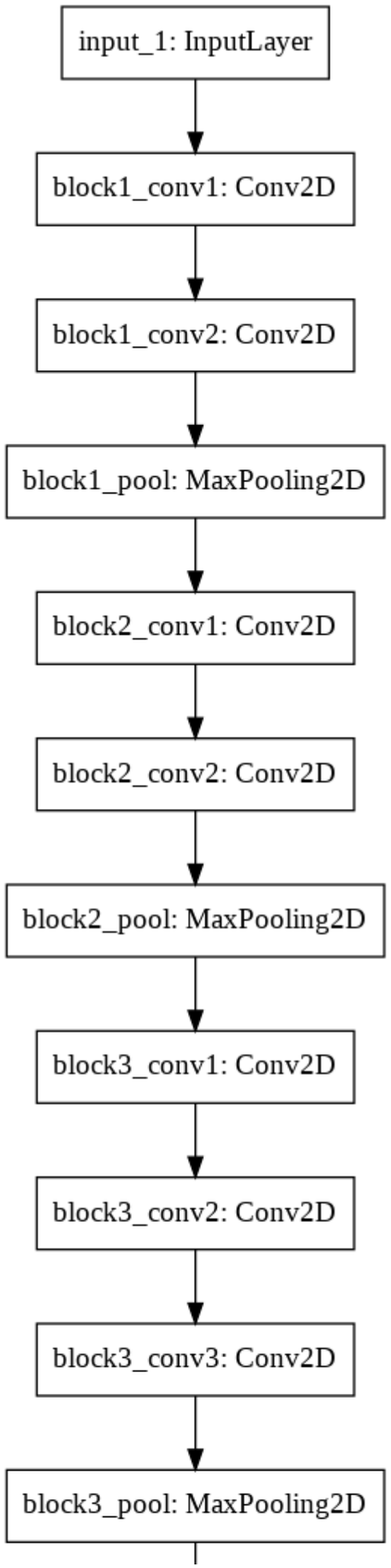
```
import tensorflow as tf
import os
import numpy as np
import pandas as pd
from keras_preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.models import Model
from tensorflow.keras import layers
from tensorflow.keras.layers import Activation,BatchNormalization
from keras.layers import Conv2D, MaxPooling2D, Flatten,Dropout, Dense
from keras import regularizers, optimizers
from keras import backend as K
from keras.applications.vgg16 import VGG16
from keras.callbacks import TensorBoard
from keras.callbacks import History
%tensorflow_version 1.x
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

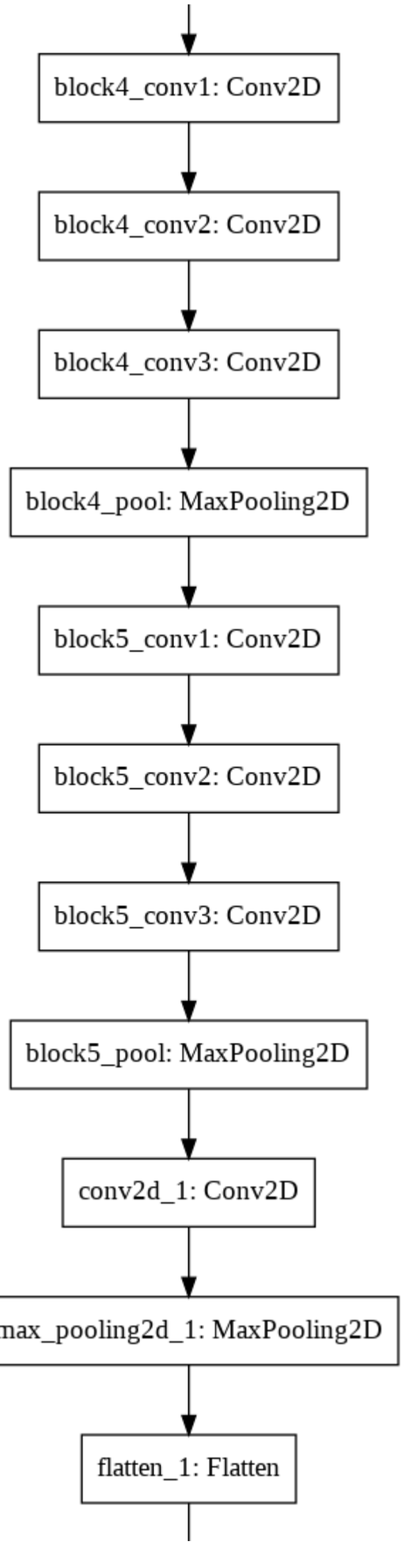
```
Using TensorFlow backend.
TensorFlow is already loaded. Please restart the runtime to change versions.
```

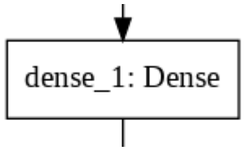
```
vgg_model = VGG16(weights='imagenet',
                  include_top=False
```

```
                              include_top=False,
                              input_shape=(224, 224, 3))

# Creating dictionary that maps layer names to the layers
layer_dict = dict([(layer.name, layer) for layer in vgg_model.layers])

# Getting output of the last VGG layer
x = layer_dict['block5_pool'].output

# Stacking a new simple convolutional network
x = Conv2D(filters=64, kernel_size=(3, 3), activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Flatten()(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(32, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(16, activation='softmax')(x)

# Creating model1.

model1 = Model(vgg_model.input,x)

# pre-trained bottom layers are not trainable
for layer in model1.layers[:19]:
    layer.trainable = False

# compile the model
model1.compile(loss='categorical_crossentropy',
                    optimizer='rmsprop',
                    metrics=['accuracy'])
# Plot model1 graph
tf.keras.utils.plot_model(model1, to_file='Model1.png')
from IPython.display import Image
Image(filename='Model1.png')
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 5s 0us/step
```




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_1: Conv2D
max_pooling2d_1: MaxPooling2D
flatten_1: Flatten

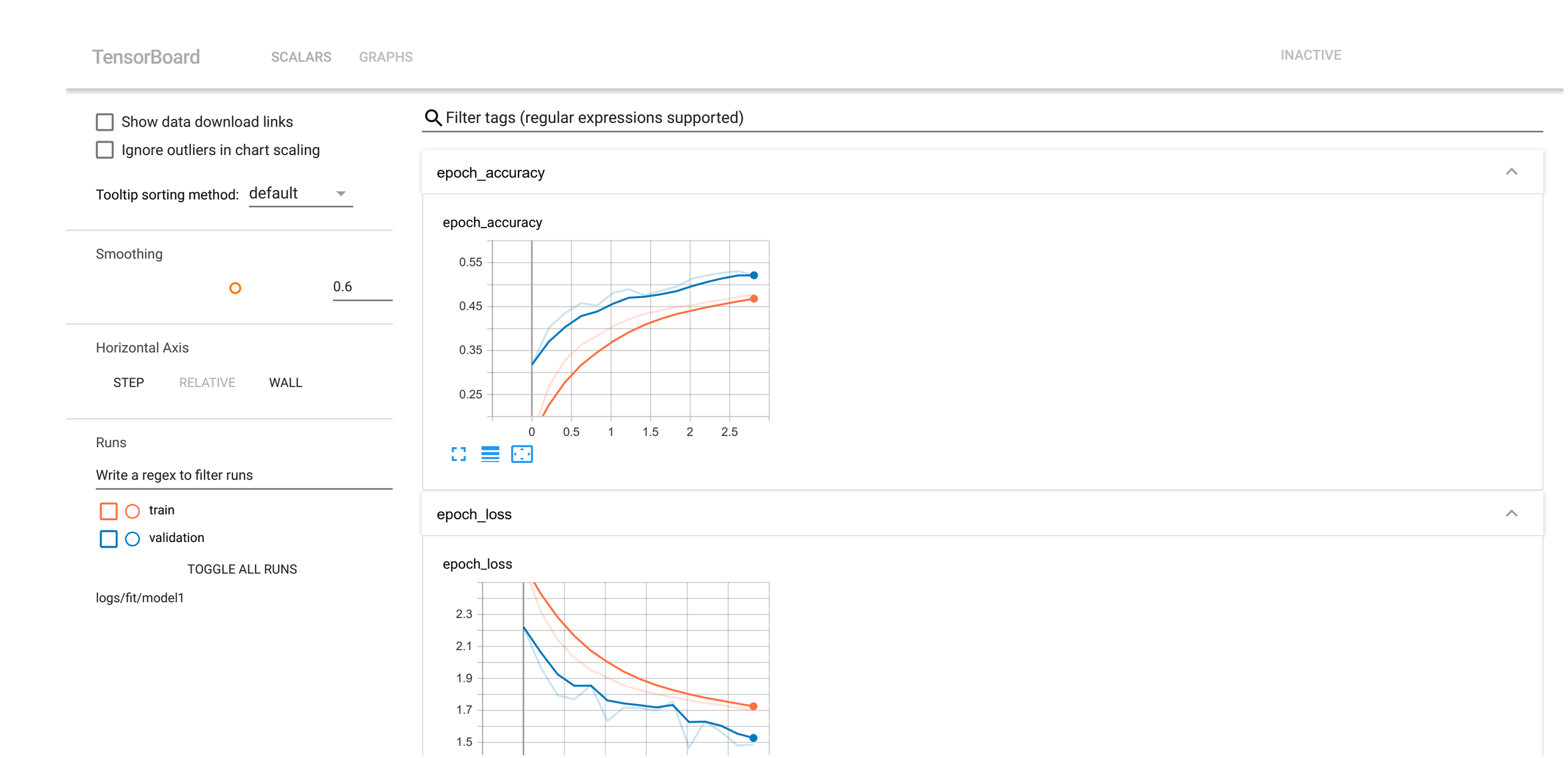



```
dir_path = "/content/data_final/"
traindf=pd.read_csv("/content/labels_final.csv",dtype=str)

def append_dir(fn):
  return "/content/data_final/"+fn

traindf["path"]=traindf["path"].apply(append_dir)
img_width, img_height = 224,224

top_model_weights_path = '/content/drive/My Drive/Colab Notebooks/Transfer learning/model1.h5'
nb_train_samples = 36000
nb_valid_samples = 12000
epochs =15
batch_size = 256

#Image generator
datagen=ImageDataGenerator(rescale=1./255.,validation_split=0.25,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

#Train generator
train_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory=None,
x_col="path",
y_col="label",
subset="training",
batch_size=256,
seed=42,
shuffle=False,
class_mode="categorical",
target_size=(224,224))

#Validation generator
valid_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory=None,
x_col="path",
y_col="label",
subset="validation",
batch_size=256,
seed=42,
shuffle=False,
class_mode="categorical",
target_size=(224,224))

##Checking time taken to fit.
import time
start = time.time()
print("model1 Fit start time: {}".format(start))
```

```
#tensoorflow callback
log_dir="logs/fit/model1"
tensorboard_callback = TensorBoard(log_dir=log_dir)
history = History()
callback_lst=[tensorboard_callback,history]

model1.fit_generator(generator=train_generator,
              epochs=epochs,
              steps_per_epoch =nb_train_samples//batch_size,
              validation_data=valid_generator,validation_steps=nb_valid_samples//batch_size,callbacks=callback_lst)
model1.save_weights(top_model_weights_path)

end = time.time()
duration = end-start
print("model1 total fit time {} ".format(duration))
```

```
Found 36000 validated image filenames belonging to 16 classes.
Found 12000 validated image filenames belonging to 16 classes.
model1 Fit start time: 1589340319.678182
Epoch 1/15
140/140 [==============================] - 768s 5s/step - loss: 2.6154 - accuracy: 0.1554 - val_loss: 2.2225 - val_accuracy: 0.3180
Epoch 2/15
140/140 [==============================] - 766s 5s/step - loss: 2.3161 - accuracy: 0.2684 - val_loss: 1.9661 - val_accuracy: 0.4013
Epoch 3/15
140/140 [==============================] - 740s 5s/step - loss: 2.1410 - accuracy: 0.3282 - val_loss: 1.7937 - val_accuracy: 0.4353
Epoch 4/15
140/140 [==============================] - 732s 5s/step - loss: 2.0286 - accuracy: 0.3629 - val_loss: 1.7691 - val_accuracy: 0.4580
Epoch 5/15
140/140 [==============================] - 731s 5s/step - loss: 1.9521 - accuracy: 0.3837 - val_loss: 1.8559 - val_accuracy: 0.4525
Epoch 6/15
140/140 [==============================] - 721s 5s/step - loss: 1.9054 - accuracy: 0.4054 - val_loss: 1.6345 - val_accuracy: 0.4810
Epoch 7/15
140/140 [==============================] - 719s 5s/step - loss: 1.8562 - accuracy: 0.4212 - val_loss: 1.7170 - val_accuracy: 0.4897
Epoch 8/15
140/140 [==============================] - 723s 5s/step - loss: 1.8264 - accuracy: 0.4333 - val_loss: 1.7148 - val_accuracy: 0.4760
Epoch 9/15
140/140 [==============================] - 711s 5s/step - loss: 1.8021 - accuracy: 0.4413 - val_loss: 1.6992 - val_accuracy: 0.4856
Epoch 10/15
140/140 [==============================] - 720s 5s/step - loss: 1.7827 - accuracy: 0.4492 - val_loss: 1.7556 - val_accuracy: 0.4955
Epoch 11/15
140/140 [==============================] - 712s 5s/step - loss: 1.7629 - accuracy: 0.4531 - val_loss: 1.4685 - val_accuracy: 0.5135
Epoch 12/15
140/140 [==============================] - 710s 5s/step - loss: 1.7455 - accuracy: 0.4605 - val_loss: 1.6319 - val_accuracy: 0.5210
Epoch 13/15
140/140 [==============================] - 710s 5s/step - loss: 1.7339 - accuracy: 0.4655 - val_loss: 1.5648 - val_accuracy: 0.5271
Epoch 14/15
140/140 [==============================] - 711s 5s/step - loss: 1.7150 - accuracy: 0.4721 - val_loss: 1.4801 - val_accuracy: 0.5307
Epoch 15/15
140/140 [==============================] - 703s 5s/step - loss: 1.6991 - accuracy: 0.4772 - val_loss: 1.4871 - val_accuracy: 0.5217
model1 total fit time 10880.527322292328
```

```
#launch the tensor board
%tensorboard --logdir logs/fit/model1
```




```
list_of_loss=history.history['loss']
list_of_val_loss=history.history['val_loss']
accuracy_lst=history.history['accuracy']
accuracy_valid_lst=history.history['val_accuracy']

epoch_count = range(1, len(list_of_val_loss) + 1)

plt.plot(epoch_count, list_of_loss, 'r--')
plt.plot(epoch_count, list_of_val_loss, 'b-')
plt.legend(['Training Loss', 'Validation Loss'])
plt.ylabel('Model1 Loss')
plt.show();

plt.plot(epoch_count, accuracy_lst, 'r--')
plt.plot(epoch_count, accuracy_valid_lst, 'b-')
plt.legend(['Training accuracy', 'Validation accuracy'])
plt.xlabel('Epoch')
```

```
plt.ylabel('Model accuracy')
plt.show();
```


Observation Model 1

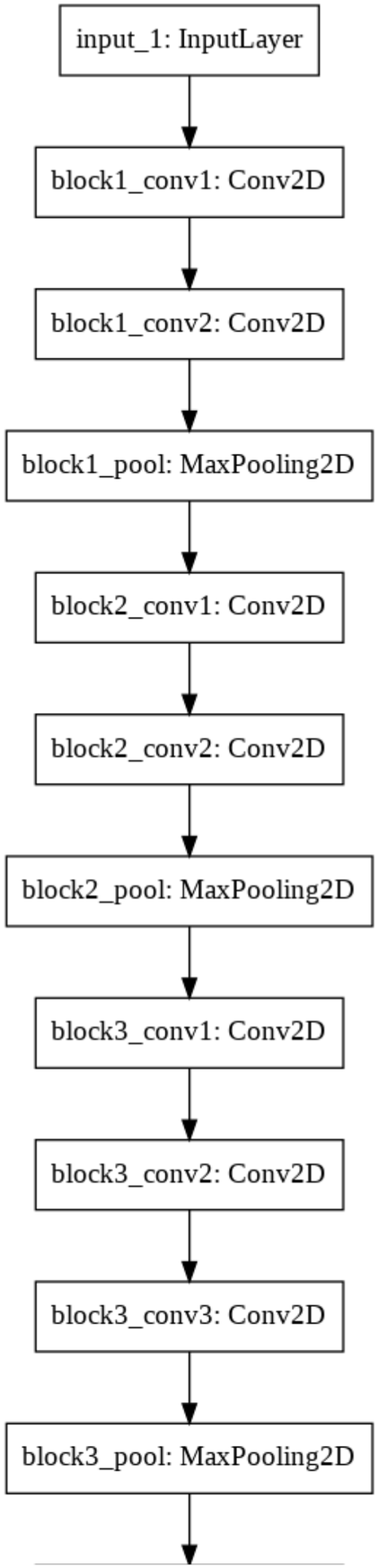
Model1 -VGG16 Top layers + simple image classifier model (1 conv layer + maxpool layer + output layer)

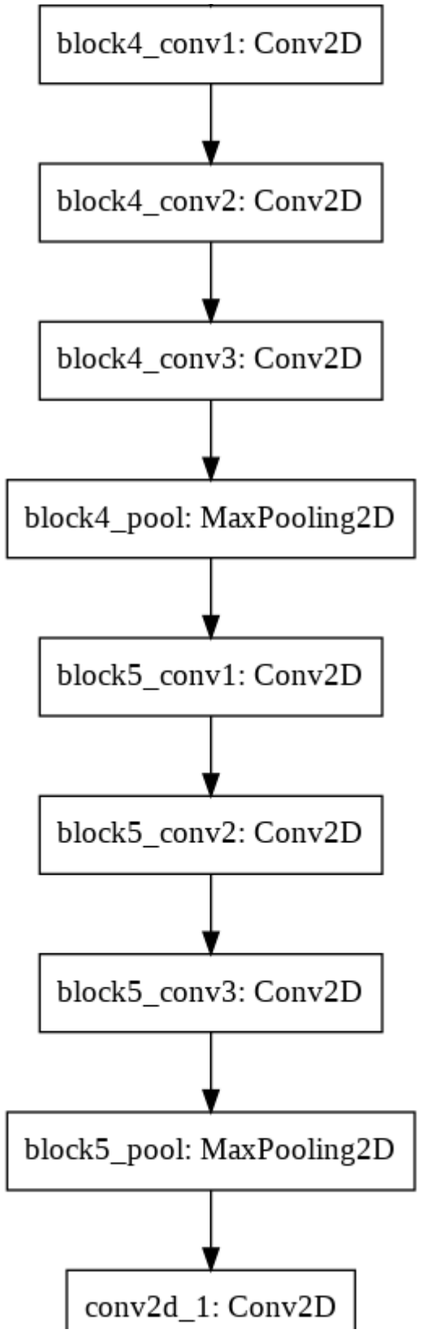
- The models accuracy and loss improves with every epoch,the model performs compariteveley well for this 16 class image classification as we are using the output of the entire VGG16 model which is primarily trained for image classifiction with huge data.

Model 2

```
vgg_model = VGG16(weights='imagenet',
                              include_top=False,
                              input_shape=(224, 224, 3))

# Creating dictionary that maps layer names to the layers
layer_dict = dict([(layer.name, layer) for layer in vgg_model.layers])

# Getting output of the last VGG layer
x = layer_dict['block5_pool'].output

# Stacking conv layers similar to dense layers
x = Conv2D(filters=4096, kernel_size=(7, 7), activation='relu')(x)
x = Conv2D(filters=4096, kernel_size=(1, 1), activation='relu')(x)
```

```
x = Flatten()(x)
x = Dense(16, activation='softmax')(x)

# Creating model2.

model2 = Model(vgg_model.input,x)

# pre-trained bottom layers are not trainable
for layer in model2.layers[:19]:
    layer.trainable = False

# compile the model
model2.compile(loss='categorical_crossentropy',
                    optimizer='rmsprop',
                    metrics=['accuracy'])
# Plot model2 graph
tf.keras.utils.plot_model(model1, to_file='Model2.png')
from IPython.display import Image
Image(filename='Model2.png')
```


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

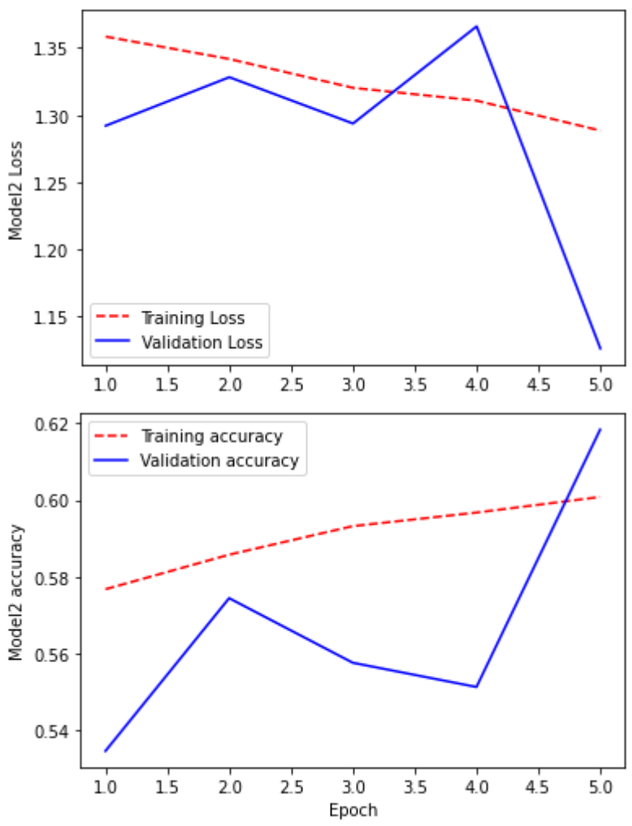



```
dir_path = "/content/data_final/"
traindf=pd.read_csv("/content/labels_final.csv",dtype=str)

def append_dir(fn):
  return "/content/data_final/"+fn

traindf["path"]=traindf["path"].apply(append_dir)

img_width, img_height = 224,224

top_model_weights_path = '/content/drive/My Drive/Colab Notebooks/Transfer learning/model2.h5'
nb_train_samples = 36000
nb_valid_samples = 12000
epochs =5
```

```
epochs =5
batch_size = 128

#Image generator
datagen=ImageDataGenerator(rescale=1./255.,validation_split=0.25,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

#Train generator
train_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory=None,
x_col="path",
y_col="label",
subset="training",
batch_size=128,
seed=42,
shuffle=False,
class_mode="categorical",
target_size=(224,224))

#Validation generator
valid_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory=None,
x_col="path",
y_col="label",
subset="validation",
batch_size=128,
seed=42,
shuffle=False,
class_mode="categorical",
target_size=(224,224))

##Checking time taken to fit.
import time
start = time.time()
print("model2 Fit start time: {}".format(start))

#tensoorflow callback
log_dir="logs/fit/model2"
tensorboard_callback = TensorBoard(log_dir=log_dir)
history = History()
callback_lst=[tensorboard_callback,history]

model2.fit_generator(generator=train_generator,
              epochs=epochs,
              steps_per_epoch =nb_train_samples//batch_size,
              validation_data=valid_generator,validation_steps=nb_valid_samples//batch_size,callbacks=callback_lst)
model2.save_weights(top_model_weights_path)

end = time.time()
duration = end-start
print("model2 total fit time {} ".format(duration))
```

```
Found 36000 validated image filenames belonging to 16 classes.
Found 12000 validated image filenames belonging to 16 classes.
model2 Fit start time: 1589362385.6134155
Epoch 1/5
281/281 [==============================] - 751s 3s/step - loss: 1.3585 - accuracy: 0.5767 - val_loss: 1.2921 - val_accuracy: 0.5345
Epoch 2/5
281/281 [==============================] - 742s 3s/step - loss: 1.3426 - accuracy: 0.5857 - val_loss: 1.3282 - val_accuracy: 0.5744
Epoch 3/5
281/281 [==============================] - 743s 3s/step - loss: 1.3207 - accuracy: 0.5932 - val_loss: 1.2937 - val_accuracy: 0.5575
Epoch 4/5
281/281 [==============================] - 740s 3s/step - loss: 1.3109 - accuracy: 0.5967 - val_loss: 1.3660 - val_accuracy: 0.5512
Epoch 5/5
281/281 [==============================] - 726s 3s/step - loss: 1.2894 - accuracy: 0.6008 - val_loss: 1.1264 - val_accuracy: 0.6183
model2 total fit time 3711.862587451935


#launch the tensor board
%tensorboard --logdir logs/fit/model2
```

```
list_of_loss=history.history['loss']
list_of_val_loss=history.history['val_loss']
accuracy_lst=history.history['accuracy']
accuracy_valid_lst=history.history['val_accuracy']

epoch_count = range(1, len(list_of_val_loss) + 1)

plt.plot(epoch_count, list_of_loss, 'r--')
plt.plot(epoch_count, list_of_val_loss, 'b-')
plt.legend(['Training Loss', 'Validation Loss'])
plt.ylabel('Model2 Loss')
plt.show();

plt.plot(epoch_count, accuracy_lst, 'r--')
plt.plot(epoch_count, accuracy_valid_lst, 'b-')
plt.legend(['Training accuracy', 'Validation accuracy'])
plt.xlabel('Epoch')
plt.ylabel('Model2 accuracy')
plt.show();
```



Obsevation Model 2

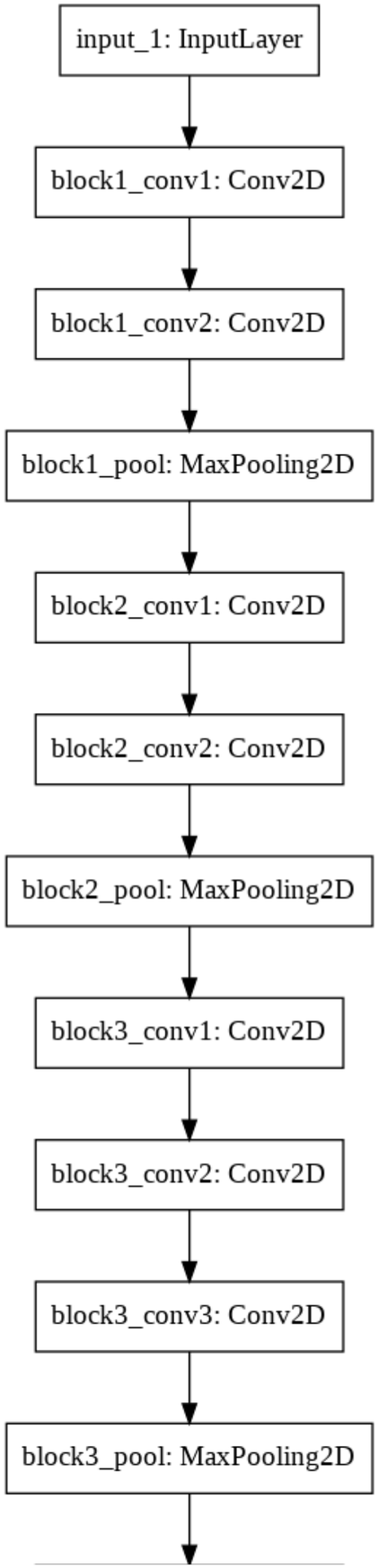
Model1 -VGG16 Top layers + simple image classifier model (1 conv layer + maxpool layer + output layer)

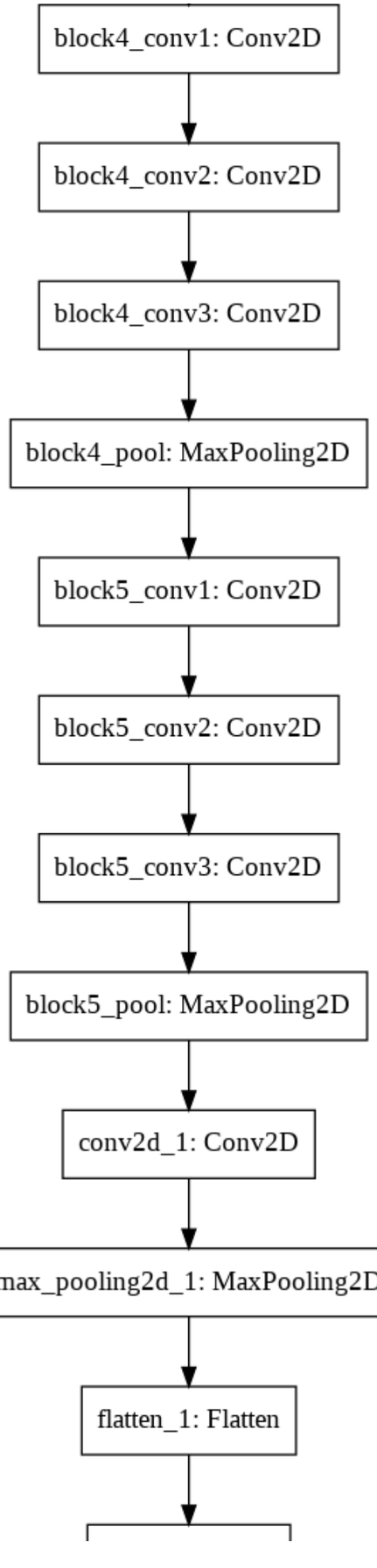
Model2 - VGG16 Top layers + dense layers in VGG16 transformed to Conv layer

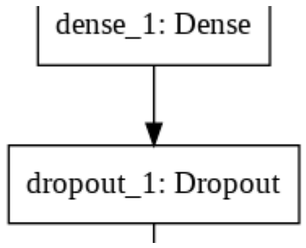
- model2 accuracy and loss improves with every epoch,model2 performs better than model 1 for this 16 class image classification,in comparitely less number of epocs we are able to achieve good accuracy, as we are only fine tuning the the output of the entire VGG16 model which is primarily trained for image classifiction with huge data

Model 3

```
vgg_model = VGG16(weights='imagenet',
                              include_top=False,
                              input_shape=(224, 224, 3))

# Creating dictionary that maps layer names to the layers
layer_dict = dict([(layer.name, layer) for layer in vgg_model.layers])

# Getting output of the last VGG layer
x = layer_dict['block5_pool'].output

# Stacking conv layers similar to dense layers
x = Conv2D(filters=4096, kernel_size=(7, 7), activation='relu')(x)
x = Conv2D(filters=4096, kernel_size=(1, 1), activation='relu')(x)
x = Flatten()(x)
x = Dense(16, activation='softmax')(x)

# Creating model3.
model3 = Model(vgg_model.input,x)

# pre-trained bottom layers are not trainable
for layer in model3.layers[:13]:
    layer.trainable = False

# compile the model
model3.compile(loss='categorical_crossentropy',
                    optimizer='rmsprop',
                    metrics=['accuracy'])

# Plot model3 graph
tf.keras.utils.plot_model(model1, to_file='Model2.png')
from IPython.display import Image
Image(filename='Model2.png')
```


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_1: Conv2D
max_pooling2d_1: MaxPooling2D
flatten_1: Flatten



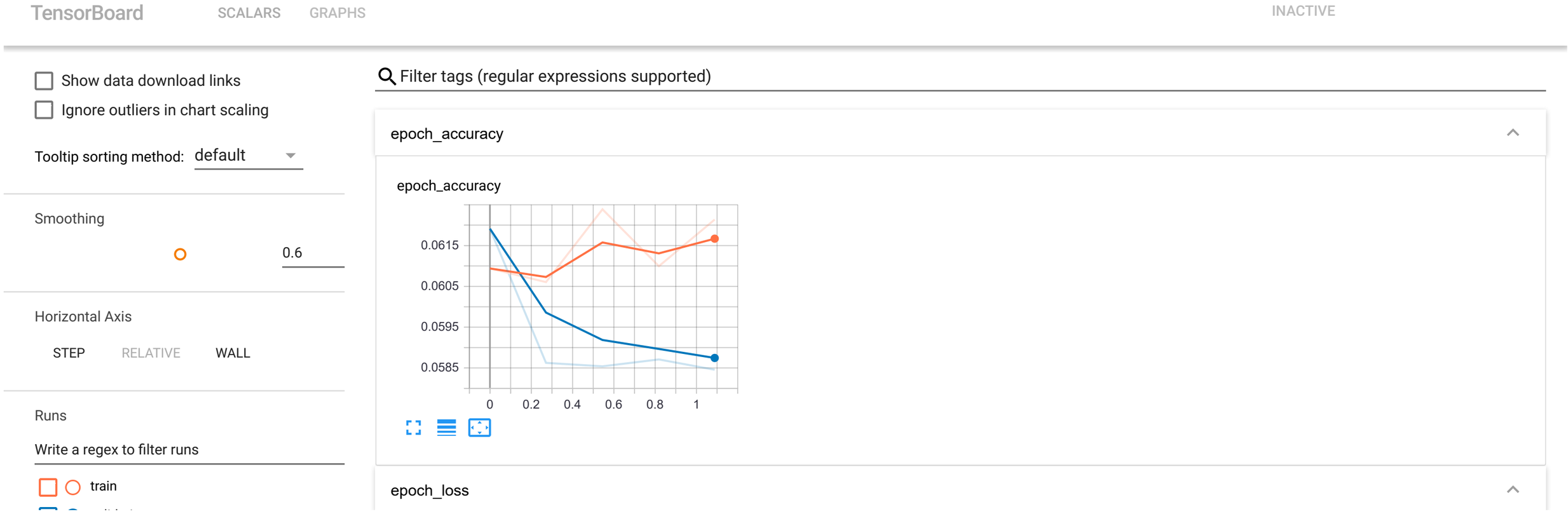

```
dir_path = "/content/data_final/"
traindf=pd.read_csv("/content/labels_final.csv",dtype=str)

def append_dir(fn):
  return "/content/data_final/"+fn

traindf["path"]=traindf["path"].apply(append_dir)


img_width, img_height = 224,224

top_model_weights_path = '/content/drive/My Drive/Colab Notebooks/Transfer learning/model3.h5'
nb_train_samples = 36000
nb_valid_samples = 12000
epochs =5
batch_size = 128

#Image generator
datagen=ImageDataGenerator(rescale=1./255.,validation_split=0.25,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

#Train generator
train_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory=None,
x_col="path",
y_col="label",
subset="training",
batch_size=128,
seed=42,
shuffle=False,
class_mode="categorical",
target_size=(224,224))

#Validation generator
valid_generator=datagen.flow_from_dataframe(
dataframe=traindf,
directory=None,
x_col="path",
y_col="label",
subset="validation",
batch_size=128,
seed=42,
shuffle=False,
class_mode="categorical",
```

```
target_size=(224,224))

##Checking time taken to fit.
import time
start = time.time()
print("model3 Fit start time: {}".format(start))

#tensoorflow callback
log_dir="logs/fit/model3"
tensorboard_callback = TensorBoard(log_dir=log_dir)
history = History()
callback_lst=[tensorboard_callback,history]

model3.fit_generator(generator=train_generator,
              epochs=epochs,
              steps_per_epoch =nb_train_samples//batch_size,
              validation_data=valid_generator,validation_steps=nb_valid_samples//batch_size,callbacks=callback_lst)
model3.save_weights(top_model_weights_path)

end = time.time()
duration = end-start
print("model3 total fit time {} ".format(duration))
```

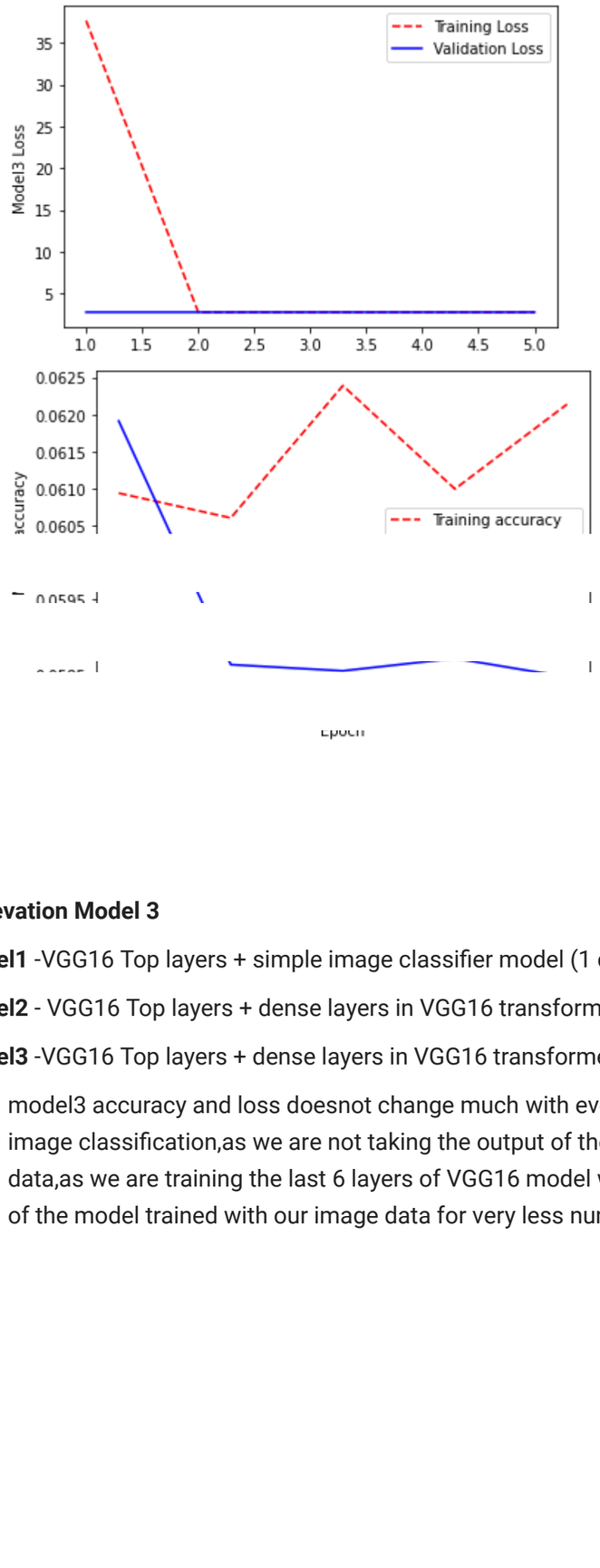
```
Found 36000 validated image filenames belonging to 16 classes.
Found 12000 validated image filenames belonging to 16 classes.
model3 Fit start time: 1589366775.1317008
Epoch 1/5
281/281 [==============================] - 970s 3s/step - loss: 37.5469 - accuracy: 0.0609 - val_loss: 2.7726 - val_accuracy: 0.0619
Epoch 2/5
281/281 [==============================] - 977s 3s/step - loss: 2.7728 - accuracy: 0.0606 - val_loss: 2.7731 - val_accuracy: 0.0586
Epoch 3/5
281/281 [==============================] - 985s 4s/step - loss: 2.7727 - accuracy: 0.0624 - val_loss: 2.7731 - val_accuracy: 0.0585
Epoch 4/5
281/281 [==============================] - 983s 3s/step - loss: 2.7727 - accuracy: 0.0610 - val_loss: 2.7724 - val_accuracy: 0.0587
Epoch 5/5
281/281 [==============================] - 973s 3s/step - loss: 2.7727 - accuracy: 0.0621 - val_loss: 2.7730 - val_accuracy: 0.0585
model3 total fit time 4903.24317407608
```

```
#launch the tensor board
%tensorboard --logdir logs/fit/model3
```

```
Reusing TensorBoard on port 6008 (pid 18377), started 0:03:52 ago. (Use '!kill 18377' to kill it.)
```




```
list_of_loss=history.history['loss']
list_of_val_loss=history.history['val_loss']
accuracy_lst=history.history['accuracy']
accuracy_valid_lst=history.history['val_accuracy']

epoch_count = range(1, len(list_of_val_loss) + 1)

plt.plot(epoch_count, list_of_loss, 'r--')
plt.plot(epoch_count, list_of_val_loss, 'b-')
plt.legend(['Training Loss', 'Validation Loss'])
plt.ylabel('Model3 Loss')
plt.show();

plt.plot(epoch_count, accuracy_lst, 'r--')
plt.plot(epoch_count, accuracy_valid_lst, 'b-')
plt.legend(['Training accuracy', 'Validation accuracy'])
plt.xlabel('Epoch')
plt.ylabel('Model3 accuracy')
plt.show();
```



Obsevation Model 3

Model1 -VGG16 Top layers + simple image classifier model (1 conv layer + maxpool layer + output layer)

Model2 - VGG16 Top layers + dense layers in VGG16 transformed to Conv layer

Model3 -VGG16 Top layers + dense layers in VGG16 transformed to Conv layer

- model3 accuracy and loss doesnot change much with every epoch,model3 performs worse than model 1 and model 2 for this 16 class image classification,as we are not taking the output of the entire VGG16 model which is primarily trained for image classifiction with huge data,as we are training the last 6 layers of VGG16 model with our image data,as the output entire VGG16 model is better than the output of the model trained with our image data for very less number of epochs.

/