Transfer Learning Assignment

Download all the data in this rar_file , it contains all the data required for the assignment. When you unrar the file you'll get the files in the following format: **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
```

There is a file named as 'labels_final.csv' , it consists of two columns. First column is path which is the required path to the images and second is the class label.

```
#the dataset that you are dealing with is quite large 3.7 GB and hence there are two methods to impor
# Method 1- you can use gdown module to get the data directly from Google drive to Colab
# the syntax is as follows !gdown --id file_id , for ex - running the below cell will import the rvl-
```

```
!gdown --id 1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
```

```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` was deprec
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu
To: /content/rvl-cdip.rar
100% 4.66G/4.66G [00:46<00:00, 99.8MB/s]
```

```
# Method -2 you can also import the data using wget function
#https://www.youtube.com/watch?v=BPUfVq7RaY8
```

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

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 data.

```
#import all the required libraries
import tensorflow as tf
import os
import numpy as np
import pandas as pd

df=pd.read_csv('labels_final.csv',dtype=str)
df.head(1)
```

	path	label
0	imagesv/v/o/h/voh71d00/509132755+-2755.tif	3

```
#preprocessed file final_x_csv
#import pandas as pd
#final_df=pd.read_csv('/content/preprocessed df.csv')
```

```
# Remove three columns as index base
#final_df.drop(final_df.columns[[0]], axis = 1, inplace = True)
```

```
#final_df.head(1)
```

	path	label
0	imagesv/v/o/h/voh71d00/509132755+-2755.tif	3

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 images only during the train data. 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

Note- In the reference notebook you were dealing with jpg images, in the given dataset you are dealing with tiff images. Imagedatagenrator works with both type of images. If you want to use custom data pipeline then you have to convert your tiff images to jpg images.

4. You are free to choose Learning rate, optimizer, loss function, image augmentation, any hyperparameters. but you have to use the same architechture what we are asking below.

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

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

https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/3426/code-example-cats-vs-dogs/8/module-8-neural-networks-computer-vision-and-deep-learning

7. Do print model.summary() and draw model_plots for each of the model.

ImageDataGenerator

```
import tensorflow as tf
import os
import numpy as np
import pandas as pd
```

```
dir_path = "/content/data_final"
```

```
#os.listdir(dir_path)
```

```
for i in os.listdir(dir_path):
    print("No of Images in ",i," category is ",len(os.listdir(os.path.join(dir_path,i))))
```

```
No of Images in  imagesc  category is  1
No of Images in  imagesk  category is  1
No of Images in  imagesf  category is  1
No of Images in  imagesu  category is  1
No of Images in  imagesp  category is  1
No of Images in  imagesi  category is  1
No of Images in  imageso  category is  1
No of Images in  imagesh  category is  1
No of Images in  imagesy  category is  1
No of Images in  imagesd  category is  1
No of Images in  imagesm  category is  1
No of Images in  imagesl  category is  1
No of Images in  imagesw  category is  1
No of Images in  imagesv  category is  1
No of Images in  imagesr  category is  1
No of Images in  imagesz  category is  1
No of Images in  imagese  category is  1
No of Images in  imagesb  category is  1
No of Images in  imagesj  category is  1
No of Images in  imagest  category is  1
No of Images in  imagesg  category is  1
No of Images in  imagesa  category is  1
No of Images in  imagesq  category is  1
No of Images in  imagesn  category is  1
No of Images in  imagesx  category is  1
```

```
from keras_preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(rescale=1/224., validation_split=0.2) #image generator

#https://vijayabhaskar96.medium.com/tutorial-on-keras-imagedatagenerator-with-flow-from-dataframe-8bd
#https://stackoverflow.com/questions/42443936/keras-split-train-test-set-when-using-imagedatagenerato
#https://www.codegrepper.com/code-examples/python/keras+imagedatagenerator+train_test_split
image_generator = ImageDataGenerator(rescale=1/224, validation_split=0.2)

train_generator = image_generator.flow_from_dataframe(dataframe=df, directory="/content/data_final",
                                                      y_col="label", class_mode="categorical",
                                                      target_size=(224,224), batch_size=32,subset='trai

validation_generator = image_generator.flow_from_dataframe(dataframe=df, directory="/content/data_fin
                                                      y_col="label", class_mode="categorical",
                                                      target_size=(224,224), batch_size=32,subset='vali
```

```
Found 38400 validated image filenames belonging to 16 classes.
Found 9600 validated image filenames belonging to 16 classes.
```

```
#importing tensorflow
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten
from tensorflow.keras.models import Model
import random as rn
```

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
from keras.callbacks import Callback
from keras.callbacks import TensorBoard
```

```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./logs/
```

```
import os
import tensorflow as tf
import datetime
log_dir = os.path.join("logs",'fits', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=0,write_graph=Tr
```

Model-1

1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with
2. After VGG-16 network without FC layers, add a new Conv block (1 Conv layer and 1 Maxpooling),
3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool La**
4.Print model.summary() and plot the architecture of the model.

Reference for plotting model

```
5. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.
```

```
#https://keras.io/guides/transfer_learning/
#https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural
base_model = tf.keras.applications.vgg16.VGG16(
    weights="imagenet",  # Load weights pre-trained on ImageNet.
    input_shape=(224, 224, 3),
    include_top=False,
)  # Do not include the ImageNet classifier at the top.
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_w
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
```

```
base_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
```

```
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

```
#https://www.tensorflow.org/guide/keras/functional
#Adding custom Layers
# Freeze the base_model
base_model.trainable = False
base_model1 = base_model.output
base_model1= tf.keras.layers.Conv2D(512,kernel_size=(2,2),padding="same",activation='relu')(base_mode
#As the output of base model 7*7*512 i.e 512 kernels having 7*7 image size
#you need to match same number of kernels in new layer
#https://keras.io/api/layers/convolution_layers/convolution2d/

base_model1= tf.keras.layers.MaxPooling2D(pool_size=(2, 2), padding='same')(base_model1)
#https://keras.io/api/layers/pooling_layers/max_pooling2d/

base_model1= Flatten()(base_model1)
#Addding dense layers

#https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense
base_model1 = tf.keras.layers.Dense(100,activation='relu')(base_model1)
base_model1=tf.keras.layers.Dense(200,activation='relu')(base_model1)
output = tf.keras.layers.Dense(16, activation="softmax")(base_model1)
# creating the final model
model_1 = Model(inputs =base_model.input, outputs = output)
# compile the model
model_1.compile(loss = "categorical_crossentropy", optimizer ='Adam', metrics=["accuracy"])
```

```
model_1.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
```

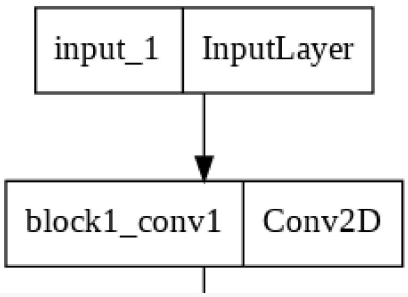
```
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 conv2d (Conv2D)             (None, 7, 7, 512)         1049088

 max_pooling2d (MaxPooling2D  (None, 4, 4, 512)        0
 )

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 100)               819300

 dense_1 (Dense)             (None, 200)               20200

 dense_2 (Dense)             (None, 16)                3216

=================================================================
Total params: 16,606,492
Trainable params: 1,891,804
```

```
train_steps = train_generator.n//train_generator.batch_size
validation_steps = validation_generator.n//validation_generator.batch_size
```

```
#fitting the model_1
model_1.fit_generator(train_generator,steps_per_epoch=train_steps, epochs=5,verbose=1,
                              validation_data=validation_generator,validation_steps=validation_steps,
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generato
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/5
1200/1200 [==============================] - 285s 225ms/step - loss: 1.2740 - accuracy: 0.6060 -
Epoch 2/5
1200/1200 [==============================] - 259s 216ms/step - loss: 0.9016 - accuracy: 0.7203 -
Epoch 3/5
1200/1200 [==============================] - 252s 210ms/step - loss: 0.7538 - accuracy: 0.7664 -
Epoch 4/5
1200/1200 [==============================] - 249s 208ms/step - loss: 0.6474 - accuracy: 0.7979 -
Epoch 5/5
1200/1200 [==============================] - 245s 204ms/step - loss: 0.5547 - accuracy: 0.8263 -
<keras.callbacks.History at 0x7f38902a9d10>
```

```
#https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model
tf.keras.utils.plot_model(
    model_1,
    to_file='model_1.png',
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir='TB',
    expand_nested=False,
    dpi=96,
    layer_range=None,
    show_layer_activations=False
)
```




```
%tensorboard --logdir logs
```

TensorBoard SCALARS GRAPHS TIME SERIES INACTIVE

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

STEP RELATIVE

WALL

Runs

Write a regex to filter runs

fits/20220518-133807/train

fits/20220518-133807/validation

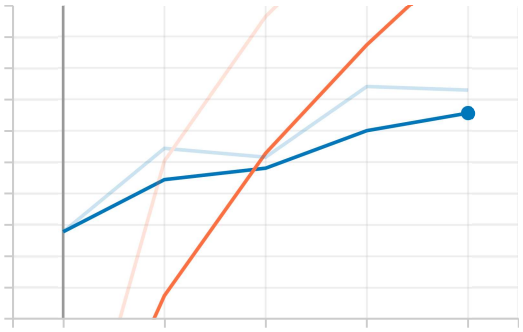
TOGGLE ALL RUNS

logs

Filter tags (regular expressions supported)

epoch_accuracy

epoch_accuracy
tag: epoch_accuracy



epoch_loss

epoch_loss
tag: epoch_loss

Observations:

1)Trainable Parameters of Model1 were reduced compared to base model, bcoz of fine tune the last layers with custom implementation of adding layers.

2)Accuracy of the model with each epoch was increased even fine tuning the last layers

Model-2

```
1. Use VGG-16 pretrained network without Fully Connected layers and initilize all the weights with
2. After VGG-16 network without FC layers, don't use FC layers, use conv layers only as Fully conne
layer can be converted to a CONV layer. This conversion will reduce the No of Trainable parameters
For example, an FC layer with K=4096 that is looking at some input volume of size 7×7×512 can be eq
In other words, we are setting the filter size to be exactly the size of the input volume, and henc
simply be 1×1×4096 since only a single depth column “fits” across the input volume, giving identica
initial FC layer. You can refer this link to better understanding of using Conv layer in place of f
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC
4. 4.Print model.summary() and plot the architecture of the model.
Reference for plotting model
5. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 net
```

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

https://www.quora.com/How-do-I-transfer-fully-connected-layer-to-fully-convolutional-layers

```
#https://www.tensorflow.org/guide/keras/functional
for layer in base_model.layers:
    layer.trainable = False
##Adding custom Layers
#model_1
base_model1 = base_model.output

base_model1= tf.keras.layers.Conv2D(4096,kernel_size=(7,7),strides=1,activation='relu')(base_model1)
#As the output of base model 7*7*512 i.e 512 kernels having 7*7 image size
#you need to match same number of kernels in new layer
#https://keras.io/api/layers/convolution_layers/convolution2d/

base_model1=tf.keras.layers.Conv2D(4096,kernel_size=(1,1),strides=1,activation='relu')(base_model1)
#https://keras.io/api/layers/pooling_layers/max_pooling2d/

base_model1= Flatten()(base_model1)
#Addding dense layers
```

```
#https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

output = tf.keras.layers.Dense(16, activation="softmax")(base_model1)
# creating the final model
model_2 = Model(inputs =base_model.input, outputs = output)
# compile the model
model_2.compile(loss = "categorical_crossentropy", optimizer ='Adam', metrics=["accuracy"])

model_2.summary()
```

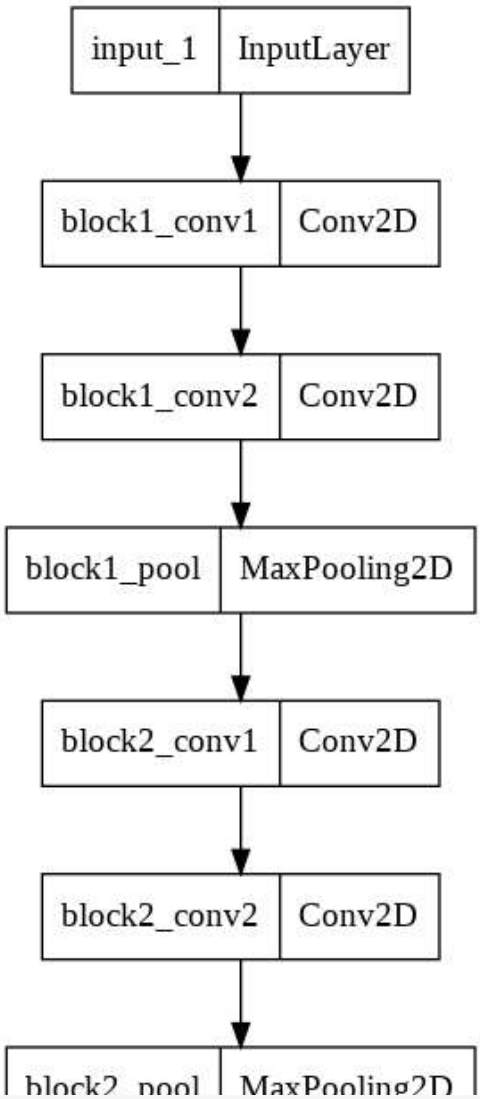
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 conv2d_2 (Conv2D)           (None, 1, 1, 4096)        102764544

 conv2d_3 (Conv2D)           (None, 1, 1, 4096)        16781312

 flatten_1 (Flatten)         (None, 4096)              0

 dense_1 (Dense)             (None, 16)                65552

=================================================================
Total params: 134,326,096
```

```
Trainable params: 119,611,408
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
#fitting the model_1
model_2.fit_generator(train_generator,steps_per_epoch=train_steps, epochs=5,verbose=1,
                              validation_data=validation_generator,validation_steps=validation_steps,
                      callbacks=[tensorboard_callback])
```

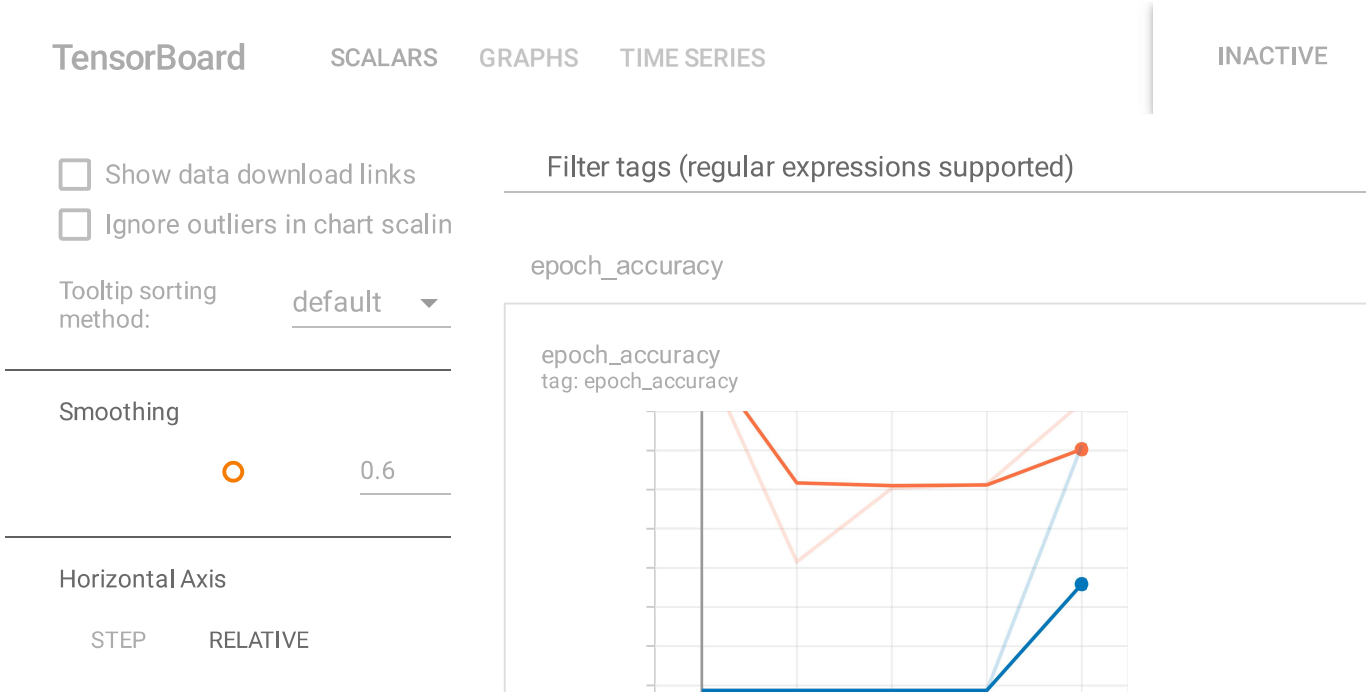
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: UserWarning: `Model.fit_generato
  after removing the cwd from sys.path.
Epoch 1/5
1200/1200 [==============================] - 415s 345ms/step - loss: 2.7728 - accuracy: 0.0628 -
Epoch 2/5
1200/1200 [==============================] - 412s 343ms/step - loss: 2.7728 - accuracy: 0.0601 -
Epoch 3/5
1200/1200 [==============================] - 413s 344ms/step - loss: 2.7728 - accuracy: 0.0610 -
Epoch 4/5
1200/1200 [==============================] - 412s 343ms/step - loss: 2.7728 - accuracy: 0.0611 -
Epoch 5/5
1200/1200 [==============================] - 413s 344ms/step - loss: 2.7728 - accuracy: 0.0621 -
<keras.callbacks.History at 0x7fc90fd04cd0>
```

```
tf.keras.utils.plot_model(
    model_2,
    to_file='model_2.png',
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir='TB',
    expand_nested=False,
    dpi=96,
    layer_range=None,
    show_layer_activations=False
)
```




```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 256), started 0:36:10 ago. (Use '!kill 256' to kill it.)
```



Observations:

1)Number of trainable parameters of Model_2 were increased compared to model_1.This is because of converting dense layers into conv layers having same number of parameters in dense layer.

2)Due to conversion of Dense into conv layers the model accuracy was decreased and computational time also increased.

fits/20220520-072253/valida epoch_loss

Model-3

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

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

```
#https://classroom.appliedroots.com/v2/faqs/vkAgx07d/
#https://www.tensorflow.org/guide/keras/functional
#Adding custom Layers
#model_1
for layer in base_model.layers[-6:]:
    layer.trainable = True
base_model1 = base_model.output
base_model1= tf.keras.layers.Conv2D(4096,kernel_size=(7,7),strides=1,activation='relu')(base_model1)
#As the output of base model 7*7*512 i.e 512 kernels having 7*7 image size
#you need to match same number of kernels in new layer
#https://keras.io/api/layers/convolution_layers/convolution2d/

base_model1=tf.keras.layers.Conv2D(4096,kernel_size=(1,1),strides=1,activation='relu')(base_model1)
#https://keras.io/api/layers/pooling_layers/max_pooling2d/
```

```
base_model1= Flatten()(base_model1)
#Addding dense layers

#https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

output = tf.keras.layers.Dense(16, activation="softmax")(base_model1)
# creating the final model
model_3 = Model(inputs =base_model.input, outputs = output)
# compile the model
model_3.compile(loss = "categorical_crossentropy", optimizer ='Adam', metrics=["accuracy"])
```

```
model_3.summary()
```

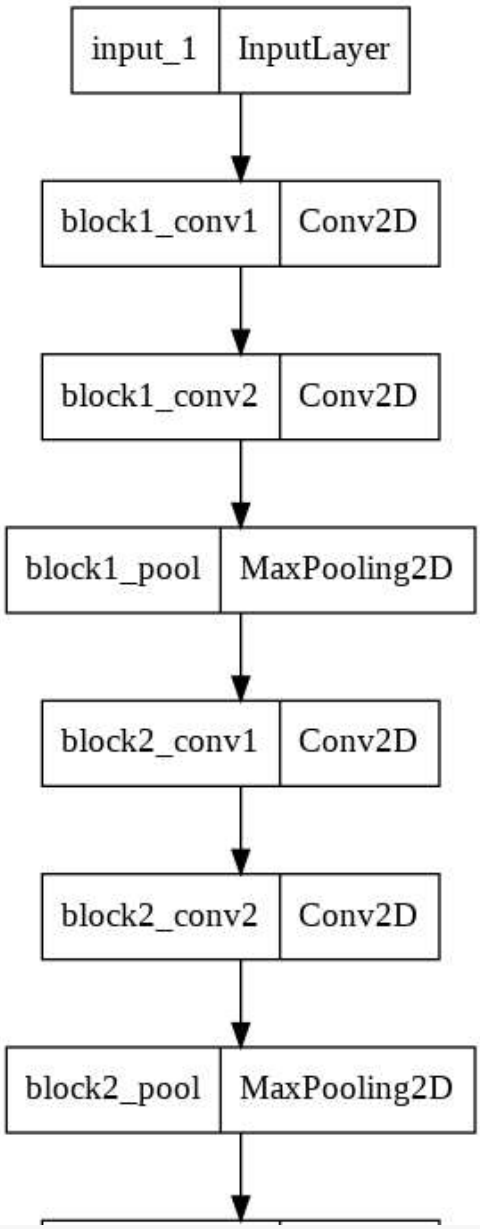
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 conv2d (Conv2D)             (None, 1, 1, 4096)        102764544

 conv2d_1 (Conv2D)           (None, 1, 1, 4096)        16781312
```

```
 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 16)                65552

=================================================================
Total params: 134,326,096
Trainable params: 134,326,096
Non-trainable params: 0
_________________________________________________________________
```

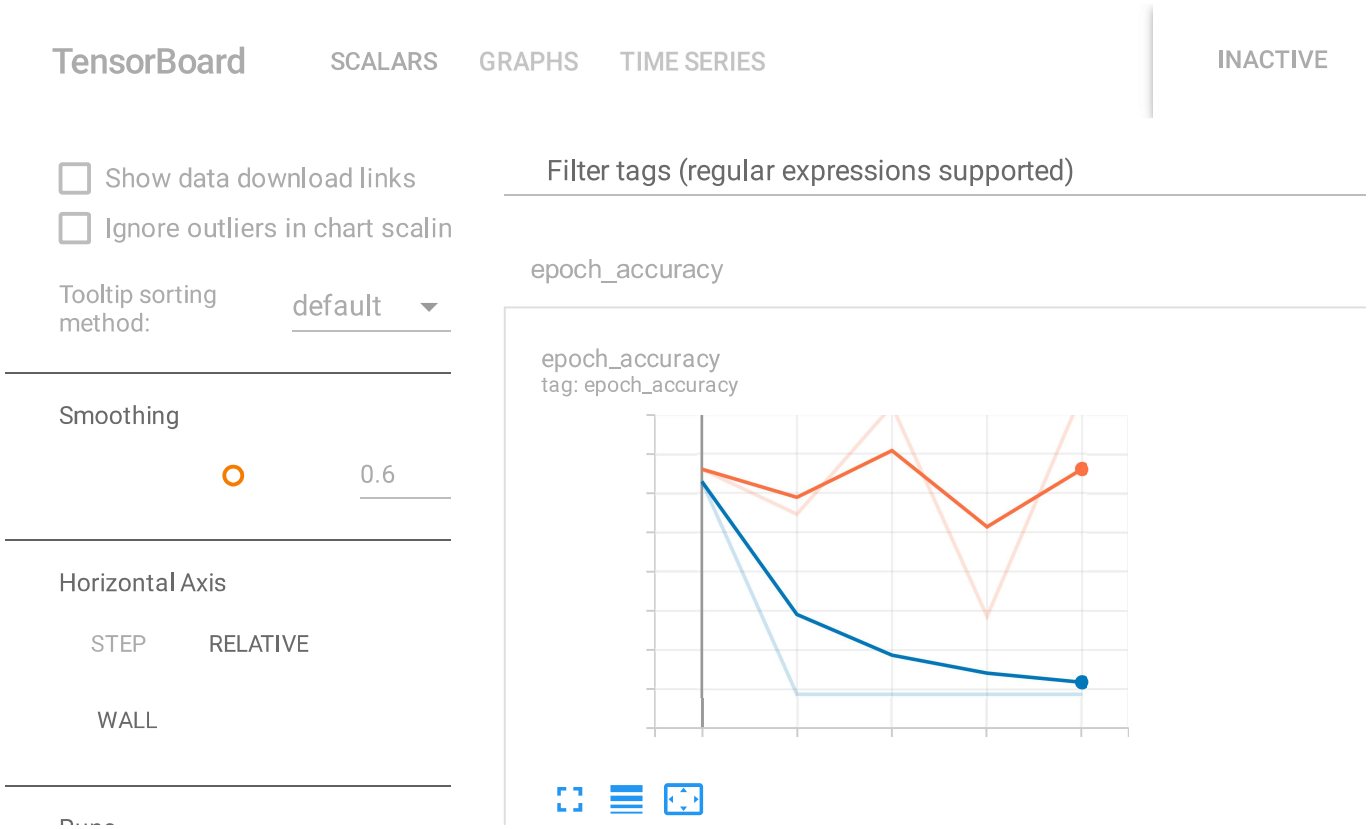
```
#fitting model_3
model_3.fit_generator(train_generator,steps_per_epoch=train_steps, epochs=5,
                              validation_data=validation_generator,validation_steps=validation_steps,
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generato
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/5
1200/1200 [==============================] - 892s 726ms/step - loss: 3.0239 - accuracy: 0.0613 -
Epoch 2/5
1200/1200 [==============================] - 878s 731ms/step - loss: 2.7728 - accuracy: 0.0607 -
Epoch 3/5
1200/1200 [==============================] - 879s 732ms/step - loss: 2.7728 - accuracy: 0.0621 -
Epoch 4/5
1200/1200 [==============================] - 878s 732ms/step - loss: 2.7728 - accuracy: 0.0594 -
Epoch 5/5
1200/1200 [==============================] - 878s 732ms/step - loss: 2.7728 - accuracy: 0.0623 -
<keras.callbacks.History at 0x7fc8920c1f50>
```

```
tf.keras.utils.plot_model(
    model_3,
    to_file='model_3.png',
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir='TB',
    expand_nested=False,
    dpi=96,
    layer_range=None,
    show_layer_activations=False
)
```




```
%tensorboard --logdir logs
```



Observations:

1)In earlier models because of the freezing of weights except top layers the trainable parameters compared to model 3 were decreased,but in model 3 itself we retrain the last 6 layers so the number of trainable parameters were increased and the accuracy of the model also decreased

2)if you want high accuracy then increase number of epochs of the model since rise in no of trainable parameters.