In [1]:
```
from google.colab import drive
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9
47318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_u
ri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2
f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2faut
h%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.google.
com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.goo
gleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=
code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%
2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2
fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.reado
nly)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [2]:
```
'drive/My Drive/21. Transfer Learning/rvl-cdip.rar'
```

In [3]:
```
Streaming output truncated to the last 5000 lines.
Creating    data_final/imagesy/y/w/v/ywv07e00                          OK
Extracting  data_final/imagesy/y/w/v/ywv07e00/2031300697.tif             95  OK
Creating    data_final/imagesy/y/w/v/ywv14c00                          OK
Extracting  data_final/imagesy/y/w/v/ywv14c00/2080388930b.tif            95  OK
Creating    data_final/imagesy/y/w/v/ywv15a00                          OK
Extracting  data_final/imagesy/y/w/v/ywv15a00/528410291+-0291.tif        95  OK
Creating    data_final/imagesy/y/w/v/ywv90a00                          OK
Extracting  data_final/imagesy/y/w/v/ywv90a00/0060014591.tif            95  OK
Creating    data_final/imagesy/y/w/v/ywv90c00                          OK
Extracting  data_final/imagesy/y/w/v/ywv90c00/13527205.tif              95  OK
Creating    data_final/imagesy/y/w/v/ywv96c00                          OK
Extracting  data_final/imagesy/y/w/v/ywv96c00/CTRCONTRACTS015375-5.tif    95  OK
Creating    data_final/imagesy/y/w/w                                   OK
Creating    data_final/imagesy/y/w/w/yww13e00                          OK
Extracting  data_final/imagesy/y/w/w/yww13e00/2061012465.tif            95  OK
Creating    data_final/imagesy/y/w/w/yww44a00                          OK
Extracting  data_final/imagesy/y/w/w/yww44a00/93807744_7748.tif         95  OK
Creating    data_final/imagesy/y/w/w/yww46d00                          OK
```

1. Download all the data in this folder https://drive.google.com/open?id=1Z4TyI7FcFVEx8qdl4jO9qxvxaqLSqoEu. it contains two file both images and labels. The label file list the images and their categories 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

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.

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-imagedatagenerator-with-flow-from-dataframe-8bd5776e45c1)

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

Start of experimnnet

In [4]:
```
import pandas as pd

data = pd.read_csv('labels_final.csv') #reading the csv file

from sklearn.model_selection import train_test_split

train_path, validation_path, train_label, validation_label = train_test_split(data['p

len(train_path)
```

Out[4]: 38400

In [5]:
```
import os
labels_dict = { 0 : 'letter',1: 'form',2: 'email',3 :'handwritten',4 :'advertisement'
                5 : 'scientific report',6 : 'scientific publication',7 :'specificatio
                9 : 'news article', 10 : 'budget', 11 : 'invoice',12 : 'presentation'
               13 : 'questionnaire', 14 : 'resume', 15 : 'memo'}

labels_dict.values()


for subfolder_name in list(labels_dict.values()):
    os.makedirs(os.path.join('train_images', subfolder_name), exist_ok=True)
```

In [6]:
```
len(labels_dict)

os.listdir('train_images')
#https://thispointer.com/python-how-to-copy-files-from-one-location-to-another-using-
import shutil
from tqdm import tqdm
for file, label in tqdm(zip(train_path, train_label)):
    shutil.copy('data_final/'+file ,'train_images/'+labels_dict[label]+'/')

for subfolder_name in list(labels_dict.values()):
    os.makedirs(os.path.join('validation_images', subfolder_name))
for file, label in tqdm(zip(validation_path, validation_label)):
    shutil.copy('data_final/'+file ,'validation_images/'+labels_dict[label]+'/')


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


import tensorflow as tf
dir_path= 'train_images'
ImageFlow = tf.keras.preprocessing.image.ImageDataGenerator()
ImageGenerator_train = ImageFlow.flow_from_directory(dir_path,target_size=(224,224),s
dir_path='validation_images'
ImageGenerator_validation = ImageFlow.flow_from_directory(dir_path,target_size=(224,2
```

```
38400it [02:00, 319.56it/s]
9600it [00:35, 268.58it/s]
```

```
No of Images in  file folder  category is  2351
No of Images in  memo  category is  2401
No of Images in  scientific publication  category is  2373
No of Images in  handwritten  category is  2413
No of Images in  invoice  category is  2397
No of Images in  specification  category is  2391
No of Images in  scientific report  category is  2379
No of Images in  letter  category is  2461
No of Images in  budget  category is  2422
No of Images in  form  category is  2407
No of Images in  advertisement  category is  2393
No of Images in  news article  category is  2392
No of Images in  resume  category is  2415
No of Images in  email  category is  2379
No of Images in  presentation  category is  2428
No of Images in  questionnaire  category is  2395
No of Images in  file folder  category is  652
No of Images in  memo  category is  595
No of Images in  scientific publication  category is  612
No of Images in  handwritten  category is  592
No of Images in  invoice  category is  595
No of Images in  specification  category is  609
No of Images in  scientific report  category is  620
No of Images in  letter  category is  554
No of Images in  budget  category is  580
No of Images in  form  category is  587
No of Images in  advertisement  category is  601
No of Images in  news article  category is  609
No of Images in  resume  category is  590
No of Images in  email  category is  614
```

In [7]:
```
log_dir="logs/fit/model-1"
```

In [8]:
```

```

Model-1

1. Use VGG-16 (https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16) 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 to classify 16 classes. You are free to choose any hyperparameters/parameters of conv block, FC layers, output layer.
3. Final architecture will be **INPUT --> VGG-16 without Top layers(FC) --> Conv Layer --> Maxpool Layer --> 2 FC layers --> Output Layer**
4. Train only new Conv block, FC layers, output layer. Don't train the VGG-16 network.

In [9]:
```
'''
    This code is refenced from the keras offical documentation page for creating Imag
'''

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
batch_size = 64
# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1./255)

tf.keras.backend.clear_session()
!rm -rf logs/
#import keras
#keras.backend.clear_session()
from tensorflow.keras.applications.vgg16 import VGG16
#import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
#nb_epochs = 2
batch_size = 64
nb_classes = len(labels_dict)

vgg16_model = VGG16(input_shape=(224,224,3),weights = 'imagenet', include_top = False
x = vgg16_model.output
x = Conv2D(64, (3, 3), padding="same", kernel_initializer=tf.keras.initializers.he_no
x = MaxPooling2D(pool_size=(2, 2))(x)
x = Flatten()(x)
x = Dense(64,  kernel_initializer=tf.keras.initializers.he_normal(seed=None), activat
x = Dense(20,  kernel_initializer=tf.keras.initializers.he_normal(seed=None), activat
predictions = Dense(16, activation = 'softmax')(x)
model = Model(inputs = vgg16_model.input, outputs = predictions)
for layers in vgg16_model.layers:
  layers.trainable = False
model.summary()

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=

model.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy',met
history = model.fit_generator(
        ImageGenerator_train,
        steps_per_epoch=38400//32,
        epochs=5,
        validation_data=ImageGenerator_validation,validation_steps= 9600//32, callbac
```

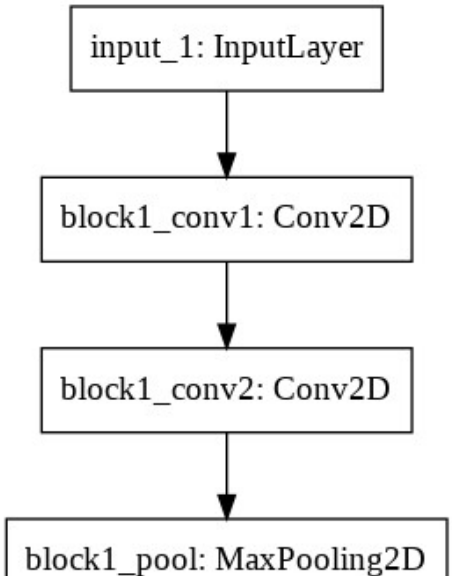
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications
/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5 (https://storage.googleap
is.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernel
s_notop.h5)
58892288/58889256 [==============================] - 1s 0us/step
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, 64)          294976
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 20)                1300
_________________________________________________________________
dense_2 (Dense)              (None, 16)                336
=================================================================
Total params: 15,048,228
Trainable params: 333,540
Non-trainable params: 14,714,688
_________________________________________________________________
WARNING:tensorflow:From <ipython-input-9-396c200b8a3e>:54: Model.fit_generator (fr
om tensorflow.python.keras.engine.training) is deprecated and will be removed in a
future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/5
1200/1200 [==============================] - 296s 247ms/step - loss: 2.7930 - accu
racy: 0.1569 - val_loss: 2.3942 - val_accuracy: 0.3074
Epoch 2/5
1200/1200 [==============================] - 286s 238ms/step - loss: 1.8877 - accu
racy: 0.4765 - val_loss: 1.6503 - val_accuracy: 0.5476
Epoch 3/5
1200/1200 [==============================] - 269s 224ms/step - loss: 1.4253 - accu
racy: 0.6131 - val_loss: 1.4141 - val_accuracy: 0.6157
Epoch 4/5
1200/1200 [==============================] - 258s 215ms/step - loss: 1.2588 - accu
racy: 0.6687 - val_loss: 1.3624 - val_accuracy: 0.6515
Epoch 5/5
```

```
1200/1200 [==============================] - 251s 209ms/step - loss: 1.1661 - accu
racy: 0.7042 - val_loss: 1.3809 - val_accuracy: 0.6596
```

In [10]:
```
from tensorflow.keras.utils import plot_model
```

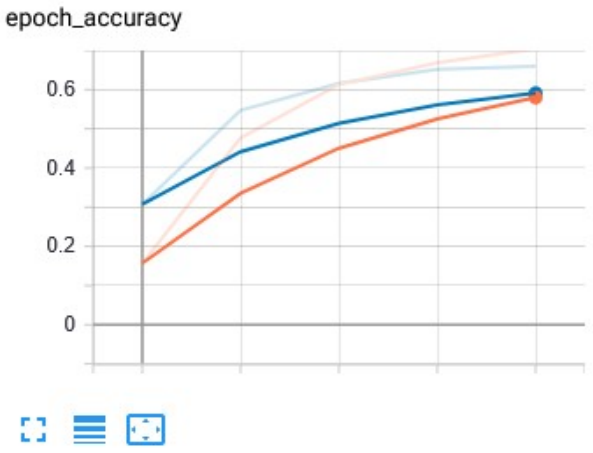
Out[10]:

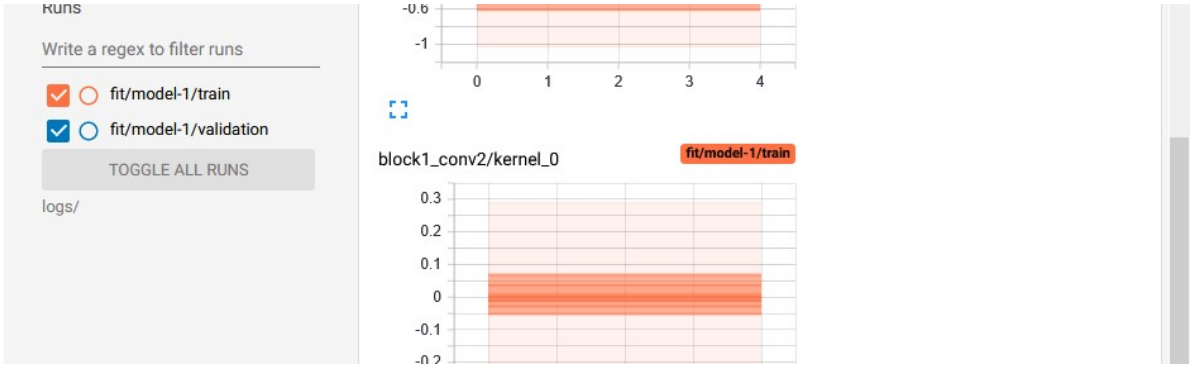


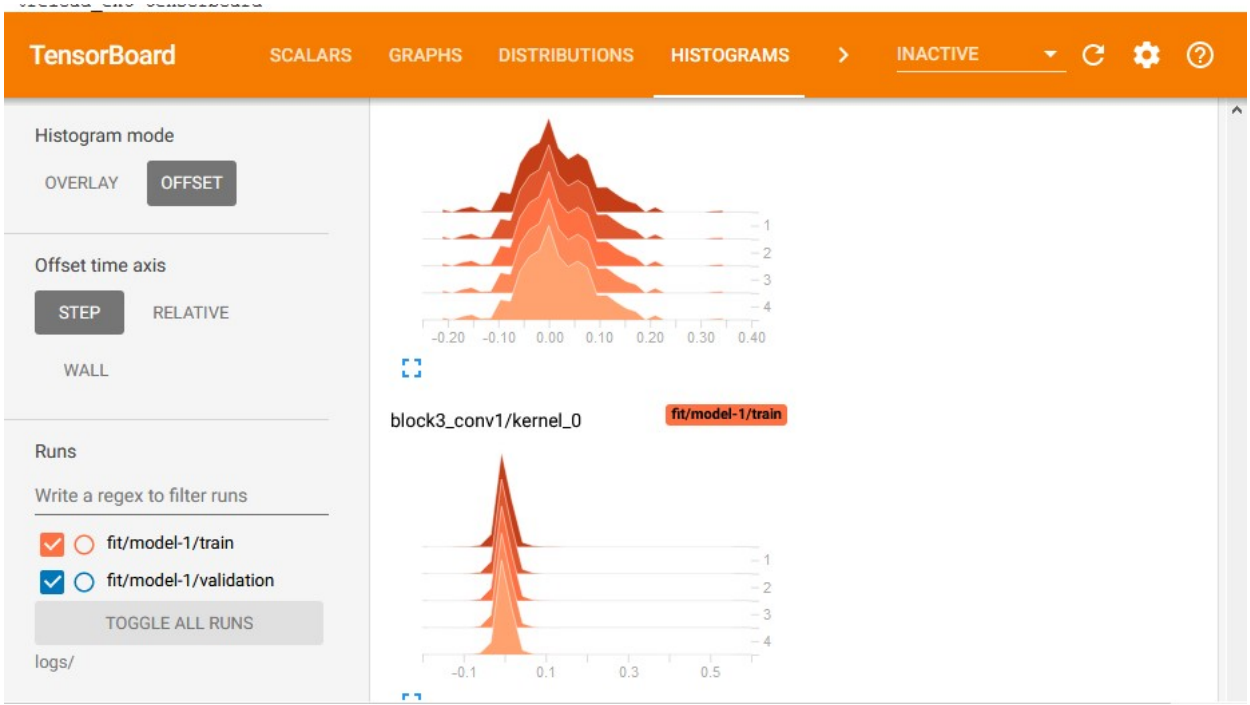
In [11]:
```
%load_ext tensorboard
# Clear any logs from previous runs
#!rm -rf ./logs/
```

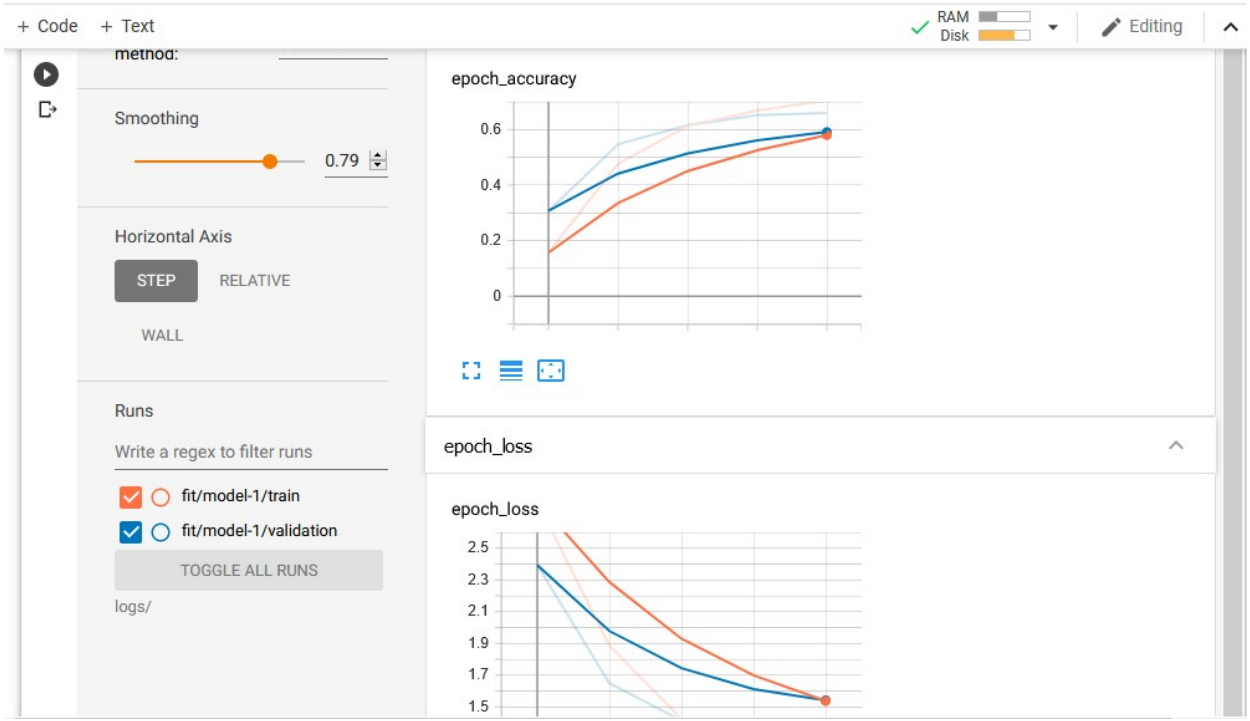
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
<IPython.core.display.Javascript object>
```




Runs
Write a regex to filter runs
fit/model-1/train
fit/model-1/validation
TOGGLE ALL RUNS
logs/
block1_conv2/kernel_0
fit/model-1/train


TensorBoard
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
Histogram mode
OVERLAY
OFFSET
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
fit/model-1/train
fit/model-1/validation
TOGGLE ALL RUNS
logs/
block3_conv1/kernel_0
fit/model-1/train


+ Code
+ Text
RAM
Disk
Editing
Smoothing
0.79
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
fit/model-1/train
fit/model-1/validation
TOGGLE ALL RUNS
logs/
epoch_accuracy
epoch_loss
epoch_loss

- 1. After each epoch model error is reducing and accuracy is increasing
- 2. Similary in the histogram we the kind ofgood distribution and seehow they are changing from each epoch and layer to layer as well.
- 3. I got the more intution on gram from here: https://stackoverflow.com/questions/42315202/understanding-tensorboard-weight-histograms

Model-2

1. Use VGG-16 (https://www.tensorflow.org/api_docs/python/tf/keras/applications/VGG16) 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 converted to a CONV layer. This conversion will reduce the No of Trainable parameters in FC layers. For example, an FC layer with K=4096 that is looking at some input volume of size 7×7×512 can be equivalently expressed as a CONV layer with F=7,P=0,S=1,K=4096. In other words, we are setting the filter size to be exactly the size of the input volume, and hence the output will simply be 1×1×4096 since only a single depth column "fits" across the input volume, giving identical result as the initial FC layer. You can refer this (http://cs231n.github.io/convolutional-networks/#convert) link to better understanding of using Conv layer in place of fully connected layers.
3. Final architecture will be VGG-16 without FC layers(without top), 2 Conv layers identical to FC layers, 1 output layer for 16 class classification. **INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer**
3. Train only last 2 Conv layers identical to FC layers, 1 output layer. Don't train the VGG-16 network.

In [12]:
```
from tensorflow.keras.applications.vgg16 import VGG16
#import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
tf.keras.backend.clear_session()
vgg16_model_2 = VGG16(input_shape=(224,224,3),weights = 'imagenet', include_top = Fal
#vgg16_model_2.summary()

for layer in vgg16_model_2.layers:
  layer.trainable = False
  print(layer.name)
```

```
input_1
block1_conv1
block1_conv2
block1_pool
block2_conv1
block2_conv2
block2_pool
block3_conv1
block3_conv2
block3_conv3
block3_pool
block4_conv1
block4_conv2
block4_conv3
block4_pool
block5_conv1
block5_conv2
block5_conv3
block5_pool
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
_________________________________________________________________
```

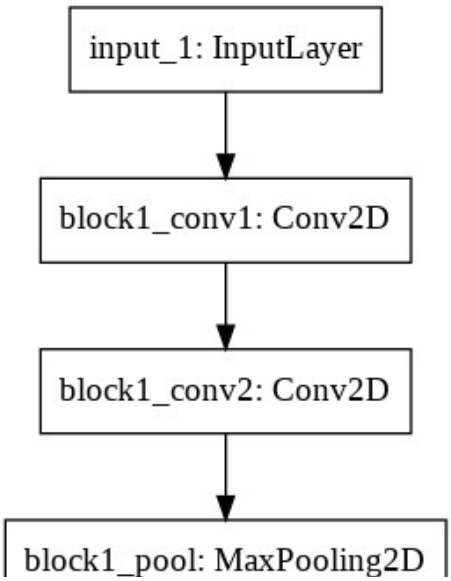
In [13]:
```
# use vgg16_model.output not vgg16_model_2.layers[-1].output
conv_layer_1 = Conv2D(4096, kernel_size=(7,7), strides=(1,1), padding='valid', kernel
conv_layer_2 = Conv2D(1000, kernel_size=(1,1), strides=(1,1), padding='valid', kernel
flatten      = Flatten()(conv_layer_2)
predictions  = Dense(16,  activation = 'softmax')(flatten)
```

```
model_2 = Model(inputs= vgg16_model_2.input, outputs = predictions)
model_2.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, 1000)        4097000
_________________________________________________________________
flatten (Flatten)            (None, 1000)              0
_________________________________________________________________
dense (Dense)                (None, 16)                16016
=================================================================
Total params: 121,592,248
Trainable params: 106,877,560
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [14]:
```
from tensorflow.keras.utils import plot_model
```

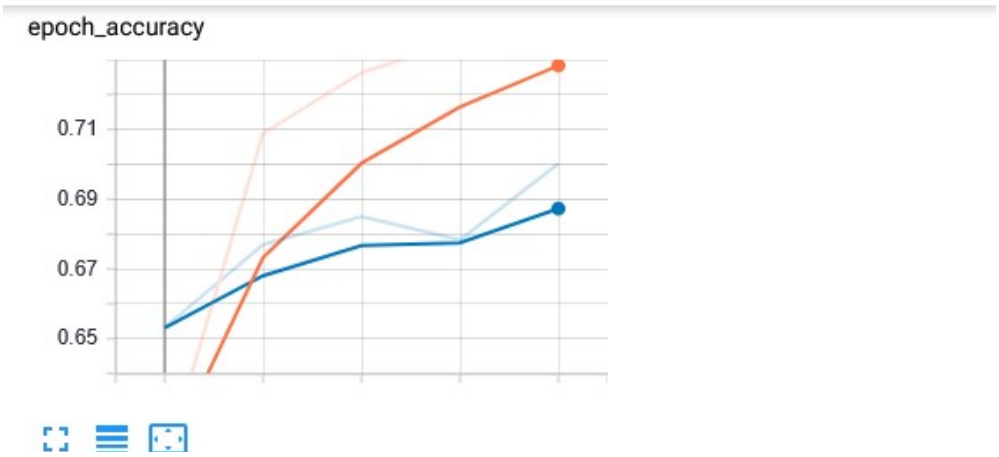
Out[14]:



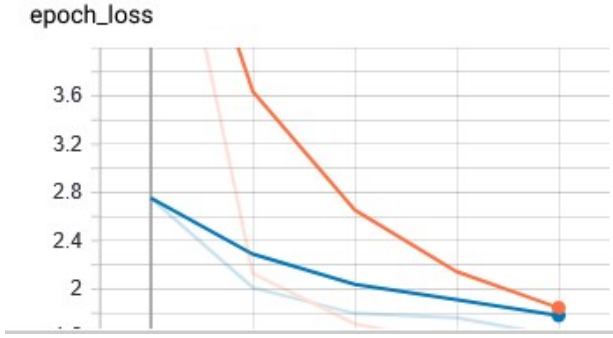

```
!rm -rf logs/
log_dir="logs/fit/model-2"
os.makedirs(log_dir, exist_ok=True)

tensorboard_callback =
tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1)


model_2.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy',m
trics=['accuracy'])
history = model_2.fit_generator(
        ImageGenerator_train,
        steps_per_epoch=38400//32,
        epochs=5,
        validation_data=ImageGenerator_validation,validation_steps= 9600//32,
callbacks=[tensorboard_callback])
```

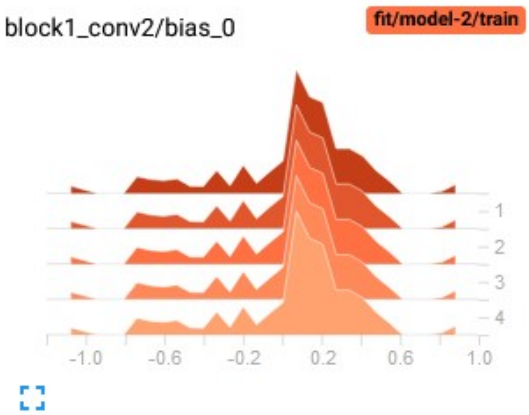
In [16]:

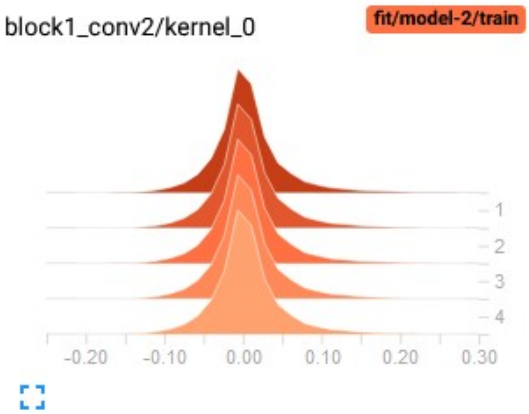
```
Reusing TensorBoard on port 6006 (pid 1159), started 0:45:49 ago. (Use '!kill 1159
' to kill it.)

<IPython.core.display.Javascript object>
```



epoch_loss


epoch_loss
3.6
3.2
2.8
2.4
2


block1_conv2/bias_0
fit/model-2/train
1
2
3
4
-1.0
-0.6
-0.2
0.2
0.6
1.0


block1_conv2/kernel_0
fit/model-2/train
1
2
3
4
-0.20
-0.10
0.00
0.10
0.20
0.30


block1_conv2/bias_0
fit/model-2/train
0.6
0.2
-0.2
-0.6
-1

- 1. After changing INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer each epoch model error is reducing and accuracy is trying to come closer
- 2. Similary in the histogram we the kind ofgood distribution and seehow they are changing from each epoch and layer to layer as well.
- 3. I got the more intution on gram from here: https://stackoverflow.com/questions/42315202/understanding-tensorboard-weight-histograms (https://stackoverflow.com/questions/42315202/understanding-tensorboard-weight-histograms)
- 4. most of the weights are in the range of -0.15 to 0.15

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 train only Last 6 Layers of VGG-16 network, 2 Conv layers identical to FC layers, 1 output layer.

In [26]:
```
from tensorflow.keras.applications.vgg16 import VGG16
#import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
tf.keras.backend.clear_session()
vgg16_model_3 = VGG16(input_shape=(224,224,3),weights = 'imagenet', include_top = Fal

number_of_last_layers_to_train = 6
non_trainable_layers_count     = len(vgg16_model_3.layers)-number_of_last_layers_to_t
for layer in vgg16_model_3.layers[:non_trainable_layers_count]:
  layer.trainable = False
  print("Non-Trainable Layer:",layer.name)
for layer in vgg16_model_3.layers[non_trainable_layers_count:]:
  layer.trainable = True
  print("Trainable Layer:",layer.name)

vgg16_model_3.summary()
```

```
Non-Trainable Layer: input_1
Non-Trainable Layer: block1_conv1
Non-Trainable Layer: block1_conv2
Non-Trainable Layer: block1_pool
Non-Trainable Layer: block2_conv1
Non-Trainable Layer: block2_conv2
Non-Trainable Layer: block2_pool
Non-Trainable Layer: block3_conv1
Non-Trainable Layer: block3_conv2
Non-Trainable Layer: block3_conv3
Non-Trainable Layer: block3_pool
Non-Trainable Layer: block4_conv1
Non-Trainable Layer: block4_conv2
Trainable Layer: block4_conv3
Trainable Layer: block4_pool
Trainable Layer: block5_conv1
Trainable Layer: block5_conv2
Trainable Layer: block5_conv3
Trainable Layer: block5_pool
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
_________________________________________________________________
```

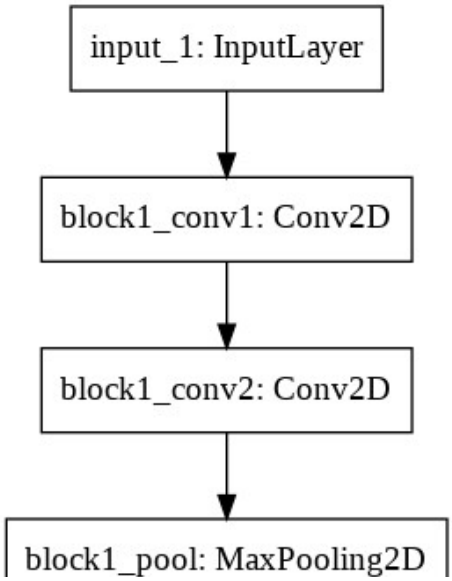
In [27]:
```
# use vgg16_model.output not vgg16_model_2.layers[-1].output
conv_layer_1 = Conv2D(4096, kernel_size=(7,7), strides=(1,1), padding='valid', kernel
```

```
conv_layer_2 = Conv2D(1000, kernel_size=(1,1), strides=(1,1), padding='valid', kernel
flatten      = Flatten()(conv_layer_2)
predictions  = Dense(16,  activation = 'softmax')(flatten)

model_3 = Model(inputs= vgg16_model_3.input, outputs = predictions)
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, 1000)        4097000
_________________________________________________________________
flatten (Flatten)            (None, 1000)              0
_________________________________________________________________
dense (Dense)                (None, 16)                16016
=================================================================
Total params: 121,592,248
Trainable params: 116,316,792
Non-trainable params: 5,275,456
_________________________________________________________________
```

In [28]:
```
from tensorflow.keras.utils import plot_model
```

Out[28]:



In [29]:
```
!rm -rf logs/
log_dir="logs/fit/model-3"
os.makedirs(log_dir, exist_ok=True)

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=


model_3.compile(optimizer=Adam(learning_rate=0.0001),loss='categorical_crossentropy',
history = model_3.fit_generator(
        ImageGenerator_train,
        steps_per_epoch=38400//64,
        epochs=3,
        validation_data=ImageGenerator_validation,validation_steps= 9600//64,
        callbacks=[tensorboard_callback])
```

```
Epoch 1/3
600/600 [==============================] - 338s 563ms/step - loss: 6.5067 - accura
cy: 0.4958 - val_loss: 4.9307 - val_accuracy: 0.6454
Epoch 2/3
600/600 [==============================] - 343s 572ms/step - loss: 4.3560 - accura
cy: 0.6787 - val_loss: 3.9051 - val_accuracy: 0.7113
Epoch 3/3
600/600 [==============================] - 343s 571ms/step - loss: 3.4507 - accura
cy: 0.7475 - val_loss: 3.2074 - val_accuracy: 0.7360
```

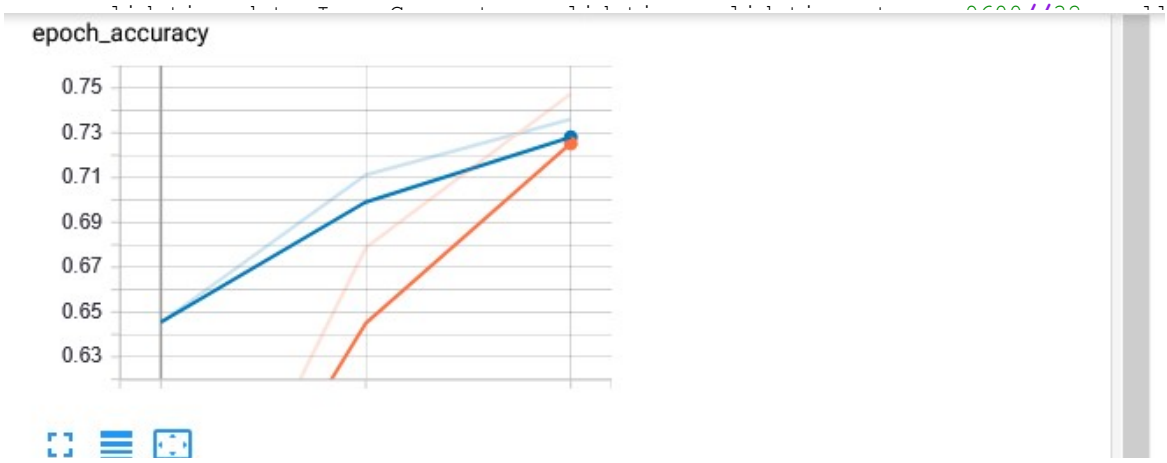
In [30]:
```
%load_ext tensorboard
```

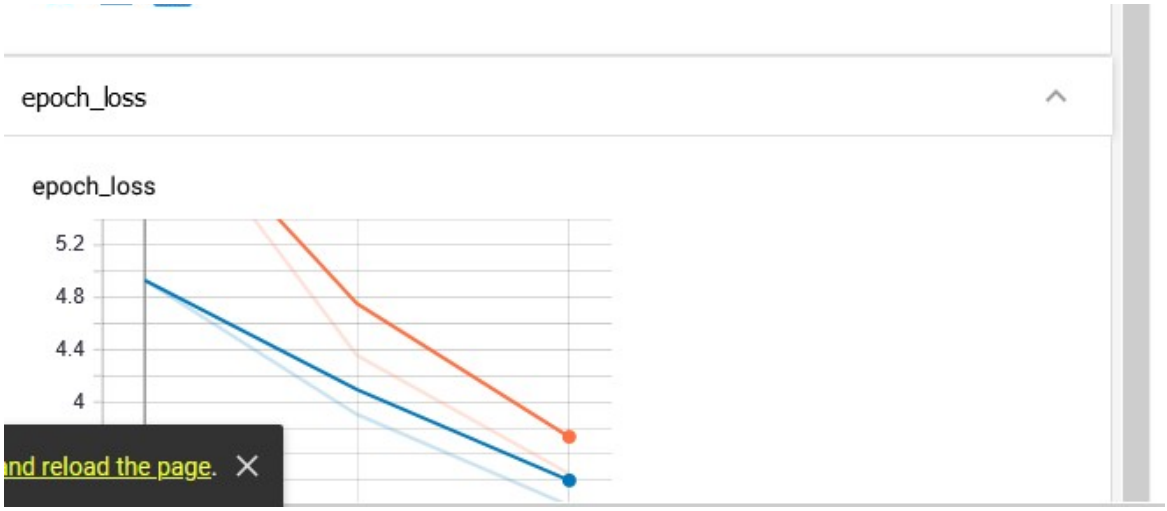
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

<IPython.core.display.Javascript object>
```

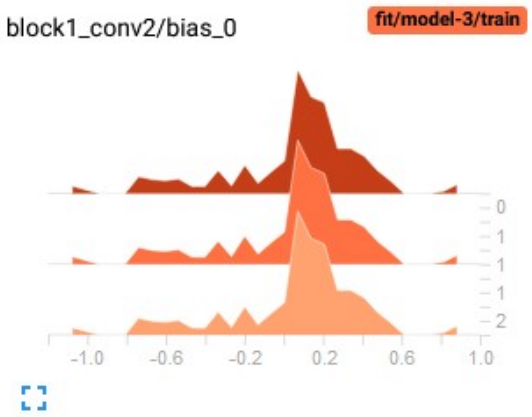
In [ ]:

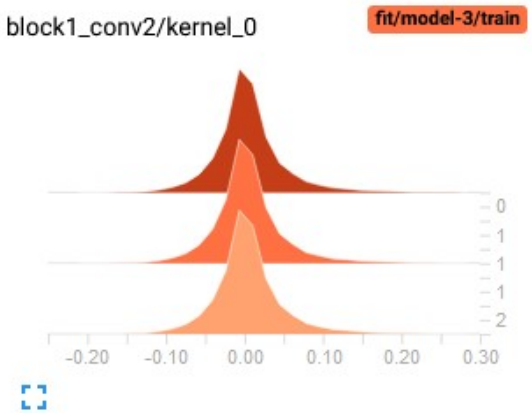
In [ ]:
```
'''
    This code is refenced from the keras offical documentation page for creating Imag
'''
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense

from tensorflow.keras.preprocessing.image import ImageDataGenerator
batch_size = 64
# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1./255)

tf.keras.backend.clear_session()

#import keras
#keras.backend.clear_session()
from tensorflow.keras.applications.vgg16 import VGG16
#import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
#nb_epochs = 2
batch_size = 64
nb_classes = len(labels_dict)

vgg16_model = VGG16(input_shape=(224,224,3),weights = 'imagenet', include_top = False
x = vgg16_model.output

conv_layer_1 = Conv2D(4096, kernel_size=(7,7), strides=(1,1), padding='valid', activa
conv_layer_2 = Conv2D(1000, kernel_size=(1,1), strides=(1,1), padding='valid', activa
flatten      = Flatten()(conv_layer_2)
predictions  = Dense(16,  activation = 'softmax')(flatten)

model_test = Model(inputs = vgg16_model.input, outputs = predictions)
for layers in vgg16_model.layers:
  layers.trainable = False
model_test.summary()

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=

model_test.compile(optimizer=Adam(learning_rate=0.001),loss='categorical_crossentropy
history = model_test.fit_generator(
        ImageGenerator_train,
        steps_per_epoch=38400//32,
        epochs=5,
```









- 1.INPUT --> VGG-16 without Top layers(FC) --> 2 Conv Layers identical to FC --> Output Layer' and train only Last 6 Layers of VGG-16 network, 2 Conv layers identical to FC layers, 1 output layer - each epoch model error is reducing and accuracy is trying to come closer
- 2. Similary in the histogram we the kind of good distribution and seehow they are changing from each epoch and layer to layer as well.
- 3. I got the more intution on gram from here: https://stackoverflow.com/questions/42315202/understanding-tensorboard-weight-histograms (https://stackoverflow.com/questions/42315202/understanding-tensorboard-weight-histograms)
- 4. it is (mostly) equally likely for a weight to have any of these values, i.e. they are (almost) uniformly distributed, most of the weights are in the range of -0.15 to 0.15