In this notebook I am going to try various pre-trained CNN models such as VGG16, InceptionV3, EfficientNet and ResNet along with a simple CNN baseline model using transfer learning method.

```
#importing libraries
import pandas as pd
import numpy as np

from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
df = pd.read_csv('/content/drive/MyDrive/Applied_ai/df_img.csv')
```

```
df.head()
```

	url	dank_or_not
0	https://i.redd.it/s044gyh084061.jpg	1
1	https://i.imgur.com/G2bnxQa.jpg	1
2	https://i.redd.it/08bjn1pan3061.jpg	1
3	https://i.redd.it/e8nptzrqs2061.jpg	1
4	https://i.redd.it/8dwnjgd962061.jpg	1

```
#saving only name of each images that was downloaded
df['url'] = df['url'].str.split("/").str[-1].str.replace('.jpg','.png')
```

```
#splitting the data
from sklearn.model_selection import train_test_split
train_df,val_df = train_test_split(df, test_size=0.2, random_state=42, shuffle=Tru
```

```
#creating train and test image generators and performing data augmentation on only
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   rotation_range=45,
                                   horizontal_flip=True,
                                   vertical_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

train_ImageGenerator = train_datagen.flow_from_dataframe(train_df, x_col = 'url',
test_ImageGenerator = test_datagen.flow_from_dataframe(val_df,x_col = 'url', y_col
```

```
    Found 3996 validated image filenames.
    Found 999 validated image filenames.
```

```
#importing libraries
import tensorflow as tf
from tensorflow.keras.utils import plot_model
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPool2D, GlobalAveragePooling2D
from tensorflow.keras.metrics import Accuracy
accuracy = Accuracy()
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
#pre-steps to load tensorboard in colab
! wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip > /dev/nu
! unzip ngrok-stable-linux-amd64.zip > /dev/null 2>&1
```

```
LOG_DIR = './log'
get_ipython().system_raw(
    'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &'
    .format(LOG_DIR)
)
```

```
get_ipython().system_raw('./ngrok http 6006 &')
```

```
! curl -s http://localhost:4040/api/tunnels | python3 -c \
    "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

```
    https://e89e956fa5f9.ngrok.io
```

Baseline CNN model

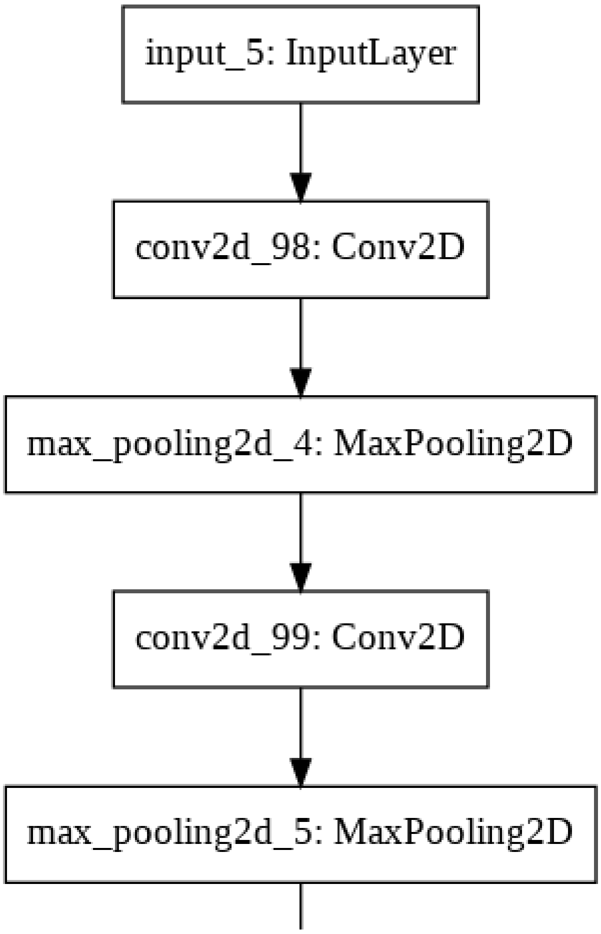
This simple model is our baseline model which will be trained using image data present.

```
#Creating baseline model with two convolution layers
input = Input(shape=(224,224,3))

conv1 = Conv2D(64, kernel_size=3, activation='relu')(input)
pool1 = MaxPool2D(pool_size=(2,2))(conv1)
```

```
pool1 = MaxPool2D(pool_size=(2,2))(conv1)

conv2 = Conv2D(128, kernel_size=3, activation='relu')(pool1)
pool2 = MaxPool2D(pool_size=(2,2))(conv2)

flat = Flatten()(pool2)
dense1 = Dense(32, activation='relu')(flat)
output = Dense(1, activation='sigmoid')(dense1)


model = Model(inputs=input, outputs=output)


print(model.summary())
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_5 (InputLayer)         [(None, 224, 224, 3)]     0         
_________________________________________________________________
conv2d_98 (Conv2D)           (None, 222, 222, 64)      1792      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 111, 111, 64)      0         
_________________________________________________________________
conv2d_99 (Conv2D)           (None, 109, 109, 128)     73856     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 54, 54, 128)       0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 373248)            0         
_________________________________________________________________
dense_7 (Dense)              (None, 32)                11943968  
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 33        
=================================================================
Total params: 12,019,649
Trainable params: 12,019,649
Non-trainable params: 0
_________________________________________________________________
None
```

```
plot_model(model)
```



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

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

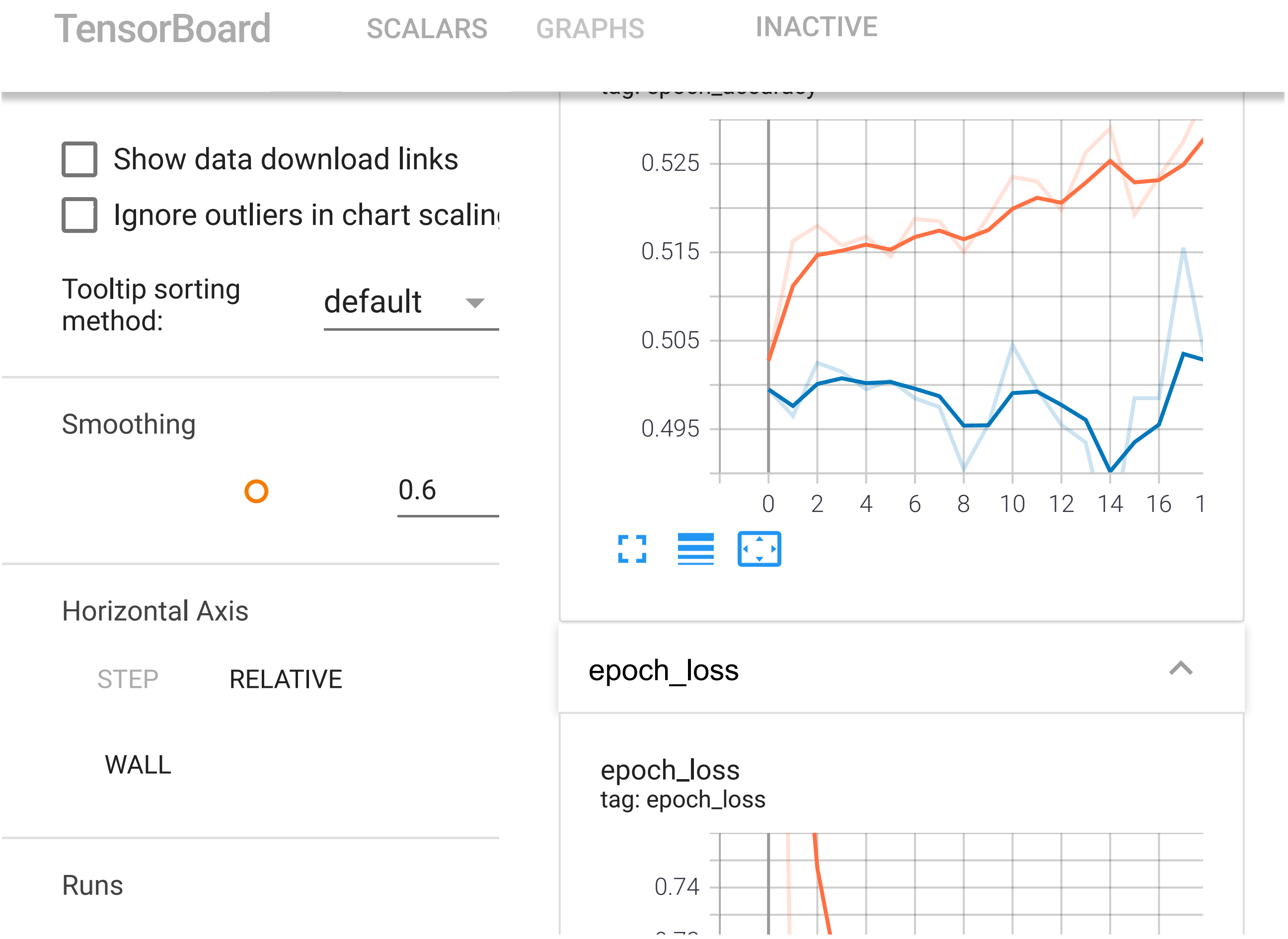
```
#tensorboard callback to draw different metrices during training
tbCallBack = tf.keras.callbacks.TensorBoard(log_dir='./log', histogram_freq=0,
                         write_graph=True,
                         write_grads=True,
                         write_images=True)
callback_list = [tbCallBack]

    WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `T
```

```
#compiling the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
#training the model
model.fit_generator(train_ImageGenerator, epochs = 20, validation_data=(test_Image

    Epoch 1/20
    125/125 [==============================] - 129s 956ms/step - loss: 0.9685 - a
    Epoch 2/20
    125/125 [==============================] - 126s 961ms/step - loss: 0.6931 - a
    Epoch 3/20
```

```
125/125 [==============================] - 141s 1s/step - loss: 0.6929 - accu
Epoch 4/20
125/125 [==============================] - 126s 941ms/step - loss: 0.6927 - a
Epoch 5/20
125/125 [==============================] - 129s 960ms/step - loss: 0.6923 - a
Epoch 6/20
125/125 [==============================] - 126s 953ms/step - loss: 0.6935 - a
Epoch 7/20
125/125 [==============================] - 126s 943ms/step - loss: 0.6920 - a
Epoch 8/20
125/125 [==============================] - 125s 943ms/step - loss: 0.6924 - a
Epoch 9/20
125/125 [==============================] - 127s 953ms/step - loss: 0.6910 - a
Epoch 10/20
125/125 [==============================] - 126s 953ms/step - loss: 0.6913 - a
Epoch 11/20
125/125 [==============================] - 126s 944ms/step - loss: 0.6943 - a
Epoch 12/20
125/125 [==============================] - 127s 948ms/step - loss: 0.6909 - a
Epoch 13/20
125/125 [==============================] - 126s 933ms/step - loss: 0.6902 - a
Epoch 14/20
125/125 [==============================] - 125s 953ms/step - loss: 0.6906 - a
Epoch 15/20
125/125 [==============================] - 140s 1s/step - loss: 0.6889 - accu
Epoch 16/20
125/125 [==============================] - 126s 947ms/step - loss: 0.6914 - a
Epoch 17/20
125/125 [==============================] - 125s 938ms/step - loss: 0.6909 - a
Epoch 18/20
125/125 [==============================] - 126s 942ms/step - loss: 0.6902 - a
Epoch 19/20
125/125 [==============================] - 127s 967ms/step - loss: 0.6893 - a
Epoch 20/20
125/125 [==============================] - 126s 941ms/step - loss: 0.6899 - a
<tensorflow.python.keras.callbacks.History at 0x7fe243dcc390>
```

```
#drawing tensorboard
%tensorboard --logdir log
```



☐ Show data download links

☐ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing ○ 0.6

Horizontal Axis

STEP RELATIVE

WALL

Runs

epoch_loss
tag: epoch_loss

```
#predicting use trained baseline model and it will predict the probabilities for t
y_pred_b = model.predict(test_ImageGenerator)
```

```
#converting probabilities to labels
y_pred_b_df = pd.DataFrame(y_pred_b).round().astype('int')
```

```
#plotting confusion matrix
cm_b = confusion_matrix(val_df['dank_or_not'], y_pred_b_df[0])
sns.heatmap(cm_b, annot=True, fmt='g')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```

```
#getting true negative, false positive, false negative and true positive values fr
tn_b, fp_b, fn_b, tp_b = cm_b.ravel()
```

```
#computing sensetivity and specificity
sensitivity_b = (tp_b/(tp_b+fn_b)).round(4)
specificity_b = (tn_b/(tn_b+fp_b)).round(4)
print('sensitivity    : ',sensitivity_b)
print('specificity    : ',specificity_b)
print('accuracy       : ',accuracy(val_df['dank_or_not'], y_pred_b_df[0]))
```

```
sensitivity    :  0.508
specificity    :  0.511
accuracy       :  tf.Tensor(0.5095095, shape=(), dtype=float32)
```

```
model.save('baseline.h5')
```

Applying VGG16

```
#importing VGG16 model
from tensorflow.keras.applications.vgg16 import VGG16
```

```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./log2/
```

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

```
#defining VGG16 model without including top fully connected layers, to perform tra
vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224,
```

```
#setting layers of VGG16 as non-trainable
for layer in vgg16_model.layers:
  layer.trainable = False
```

```
#checking layers of VGG16
for i, layer in enumerate(vgg16_model.layers):
    print(i, layer.name, layer.trainable)
```

```
0 input_4 False
1 block1_conv1 False
2 block1_conv2 False
3 block1_pool False
4 block2_conv1 False
5 block2_conv2 False
6 block2_pool False
7 block3_conv1 False
8 block3_conv2 False
```

```
9 block3_conv3 False
10 block3_pool False
11 block4_conv1 False
12 block4_conv2 False
13 block4_conv3 False
14 block4_pool False
15 block5_conv1 False
16 block5_conv2 False
17 block5_conv3 False
18 block5_pool False
```

```
vgg16_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (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: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
#adding layers after non-trainable layers of VGG16
x = vgg16_model.output
x = (Conv2D(filters = 128, kernel_size = 7, strides=1, activation='relu', kernel_i
x = (Conv2D(filters = 128, kernel_size = 1, strides=1, activation='relu', kernel_i
x = (Flatten())(x)
x = (Dense(1, activation='sigmoid'))(x)

model2 = Model(inputs=vgg16_model.input, outputs=x)


#tensorboard callback
tbCallBack2 = tf.keras.callbacks.TensorBoard(log_dir='./log2', histogram_freq=0,
                         write_graph=True,
                         write_grads=True,
                         write_images=True)
callback_list2 = [tbCallBack2]
```

```
    WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `T
```

```
#compiling the model
model2.compile(optimizer='adam',loss="binary_crossentropy", metrics=["accuracy"])


#training the model, only the layers below non-trainable layers will be trained
model2.fit_generator(train_ImageGenerator, epochs = 20, validation_data=(test_Imag
```

```
    Epoch 1/20
    125/125 [==============================] - 129s 956ms/step - loss: 0.7722 - a
    Epoch 2/20
    125/125 [==============================] - 128s 960ms/step - loss: 0.6982 - a
    Epoch 3/20
    125/125 [==============================] - 133s 960ms/step - loss: 0.6740 - a
    Epoch 4/20
    125/125 [==============================] - 129s 969ms/step - loss: 0.6823 - a
    Epoch 5/20
    125/125 [==============================] - 129s 977ms/step - loss: 0.6767 - a
    Epoch 6/20
    125/125 [==============================] - 132s 949ms/step - loss: 0.6743 - a
    Epoch 7/20
    125/125 [==============================] - 145s 1s/step - loss: 0.6647 - accu
    Epoch 8/20
    125/125 [==============================] - 127s 957ms/step - loss: 0.6753 - a
    Epoch 9/20
    125/125 [==============================] - 127s 960ms/step - loss: 0.6644 - a
    Epoch 10/20
    125/125 [==============================] - 128s 951ms/step - loss: 0.6743 - a
    Epoch 11/20
    125/125 [==============================] - 128s 967ms/step - loss: 0.6706 - a
    Epoch 12/20
    125/125 [==============================] - 133s 945ms/step - loss: 0.6710 - a
    Epoch 13/20
    125/125 [==============================] - 127s 951ms/step - loss: 0.6605 - a
    Epoch 14/20
    125/125 [==============================] - 142s 1s/step - loss: 0.6537 - accu
    Epoch 15/20
    125/125 [==============================] - 128s 958ms/step - loss: 0.6570 - a
    Epoch 16/20
```

```
125/125 [==============================] - 130s 978ms/step - loss: 0.6533 - a
Epoch 17/20
125/125 [==============================] - 142s 1s/step - loss: 0.6510 - accu
Epoch 18/20
125/125 [==============================] - 142s 1s/step - loss: 0.6532 - accu
Epoch 19/20
125/125 [==============================] - 127s 960ms/step - loss: 0.6567 - a
Epoch 20/20
125/125 [==============================] - 148s 1s/step - loss: 0.6413 - accu
<tensorflow.python.keras.callbacks.History at 0x7fe244438590>
```

```
#generating tensorboard
%tensorboard --logdir log2
```

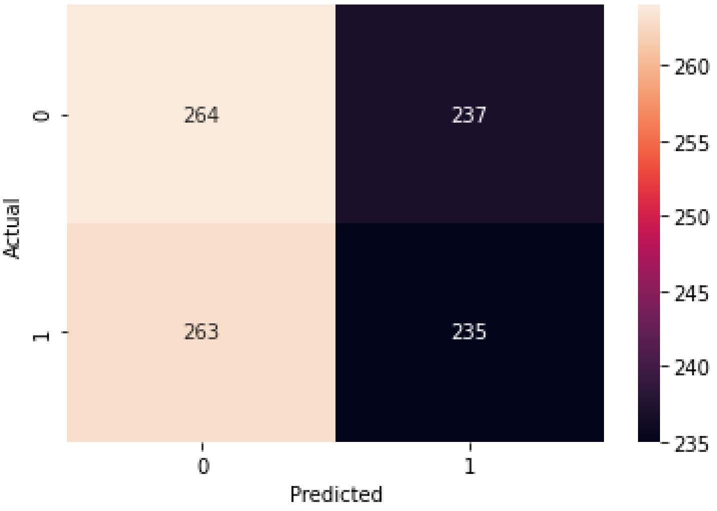
TensorBoard SCALARS GRAPHS INACTIVE

```
#predicting values for test dataset
y_pred_v = model2.predict(test_ImageGenerator)
```

```
y_pred_v_df = pd.DataFrame(y_pred_v).round().astype('int')
```

Tooltip sorting default 0.56

```
#plotting confusion matrix
cm_v = confusion_matrix(val_df['dank_or_not'], y_pred_v_df[0])
sns.heatmap(cm_v, annot=True, fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



TOGGLE ALL RUNS 0.66

```
#getting true negative, false positive, false negative and true positive values fr
tn_v, fp_v, fn_v, tp_v = cm_v.ravel()
```

```
#computing sensetivity and specificity
sensitivity_v = (tp_v/(tp_v+fn_v)).round(4)
specificity_v = (tn_v/(tn_v+fp_v)).round(4)
print('sensitivity    : ',sensitivity_v)
print('specificity    : ',specificity_v)
print('accuracy       : ',accuracy(val_df['dank_or_not'], y_pred_v_df[0]))
```

```
sensitivity    :  0.4719
specificity    :  0.5269
accuracy       :  tf.Tensor(0.50784117, shape=(), dtype=float32)
```

```
#saving the model
model2.save('vgg16_model.h5')
```

Applying Inception v3 model

```
#importing inception v3 model
from tensorflow.keras.applications.inception_v3 import InceptionV3
```

```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./log3/
```

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

```
#defining InceptionV3 model without including top fully connected layers, to perfo
inception_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
    87916544/87910968 [==============================] - 0s 0us/step
```

```
#setting layers of InceptionV3 as non-trainable
for layer in inception_model.layers:
  layer.trainable = False
```

```
#checking layers of InceptionV3
for i, layer in enumerate(inception_model.layers):
    print(i, layer.name, layer.trainable)
```

```
    249 conv2d_80 False
    250 batch_normalization_80 False
    251 activation_80 False
    252 conv2d_77 False
    253 conv2d_81 False
    254 batch_normalization_77 False
    255 batch_normalization_81 False
    256 activation_77 False
    257 activation_81 False
    258 conv2d_78 False
    259 conv2d_79 False
    260 conv2d_82 False
    261 conv2d_83 False
    262 average_pooling2d_7 False
    263 conv2d_76 False
    264 batch_normalization_78 False
    265 batch_normalization_79 False
    266 batch_normalization_82 False
    267 batch_normalization_83 False
    268 conv2d_84 False
    269 batch_normalization_76 False
    270 activation_78 False
    271 activation_79 False
    272 activation_82 False
    273 activation_83 False
    274 batch_normalization_84 False

    275 activation_76 False
    276 mixed9_0 False
    277 concatenate False
    278 activation_84 False
    279 mixed9 False
    280 conv2d_89 False
```

```
281 batch_normalization_89 False
282 activation_89 False
283 conv2d_86 False
284 conv2d_90 False
285 batch_normalization_86 False
286 batch_normalization_90 False
287 activation_86 False
288 activation_90 False
289 conv2d_87 False
290 conv2d_88 False
291 conv2d_91 False
292 conv2d_92 False
293 average_pooling2d_8 False
294 conv2d_85 False
295 batch_normalization_87 False
296 batch_normalization_88 False
297 batch_normalization_91 False
298 batch_normalization_92 False
299 conv2d_93 False
300 batch_normalization_85 False
301 activation_87 False
302 activation_88 False
303 activation_91 False
304 activation_92 False
305 batch_normalization_93 False
306 activation_85 False
307 mixed9_1 False
308 concatenate_1 False
```

```
inception_model.summary()
```

```
                                                                 activation
_________________________________________________________________________
conv2d_89 (Conv2D)              (None, 5, 5, 448)    917504      mixed9[0][
_________________________________________________________________________
batch_normalization_89 (BatchNo (None, 5, 5, 448)    1344        conv2d_89[
_________________________________________________________________________
activation_89 (Activation)      (None, 5, 5, 448)    0           batch_norm
_________________________________________________________________________
conv2d_86 (Conv2D)              (None, 5, 5, 384)    786432      mixed9[0][
_________________________________________________________________________
conv2d_90 (Conv2D)              (None, 5, 5, 384)    1548288     activation
_________________________________________________________________________
batch_normalization_86 (BatchNo (None, 5, 5, 384)    1152        conv2d_86[
_________________________________________________________________________
batch_normalization_90 (BatchNo (None, 5, 5, 384)    1152        conv2d_90[
_________________________________________________________________________
activation_86 (Activation)      (None, 5, 5, 384)    0           batch_norm
_________________________________________________________________________
activation_90 (Activation)      (None, 5, 5, 384)    0           batch_norm
_________________________________________________________________________
conv2d_87 (Conv2D)              (None, 5, 5, 384)    442368      activation
_________________________________________________________________________
conv2d_88 (Conv2D)              (None, 5, 5, 384)    442368      activation
_________________________________________________________________________
conv2d_91 (Conv2D)              (None, 5, 5, 384)    442368      activation
_________________________________________________________________________
conv2d_92 (Conv2D)              (None, 5, 5, 384)    442368      activation
_________________________________________________________________________
average_pooling2d_8 (AveragePoo (None, 5, 5, 2048)   0           mixed9[0][
_________________________________________________________________________
conv2d_85 (Conv2D)              (None, 5, 5, 320)    655360      mixed9[0][
conv2d_85 (Conv2D)              (None, 5, 5, 320)    655360      mixed8[0][
__________________________________________________________________________
batch_normalization_87 (BatchNo (None, 5, 5, 384)    1152        conv2d_87[
__________________________________________________________________________
batch_normalization_88 (BatchNo (None, 5, 5, 384)    1152        conv2d_88[
__________________________________________________________________________
batch_normalization_91 (BatchNo (None, 5, 5, 384)    1152        conv2d_91[
__________________________________________________________________________
batch_normalization_92 (BatchNo (None, 5, 5, 384)    1152        conv2d_92[
__________________________________________________________________________
conv2d_93 (Conv2D)              (None, 5, 5, 192)    393216      average_po
__________________________________________________________________________
batch_normalization_85 (BatchNo (None, 5, 5, 320)    960         conv2d_85[
__________________________________________________________________________
activation_87 (Activation)      (None, 5, 5, 384)    0           batch_norm
__________________________________________________________________________
activation_88 (Activation)      (None, 5, 5, 384)    0           batch_norm

__________________________________________________________________________
activation_91 (Activation)      (None, 5, 5, 384)    0           batch_norm
__________________________________________________________________________
activation_92 (Activation)      (None, 5, 5, 384)    0           batch_norm
__________________________________________________________________________
batch_normalization_93 (BatchNo (None, 5, 5, 192)    576         conv2d_93[
__________________________________________________________________________
activation_85 (Activation)      (None, 5, 5, 320)    0           batch_norm
__________________________________________________________________________
mixed9_1 (Concatenate)          (None, 5, 5, 768)    0           activation
                                                                 activation
```

```
#adding layers after non-trainable layers of InceptionV3
x = inception_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu', kernel_initializer='HeUniform')(x)
x = Dense(64, activation='relu', kernel_initializer='HeUniform')(x)
x = (Dense(1, activation='sigmoid'))(x)

model3 = Model(inputs=inception_model.input, outputs=x)
```

```
#tensorboard callback
tbCallBack3 = tf.keras.callbacks.TensorBoard(log_dir='./log3', histogram_freq=0,
                         write_graph=True,
                         write_grads=True,
                         write_images=True)
callback_list3 = [tbCallBack3]
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `T
```

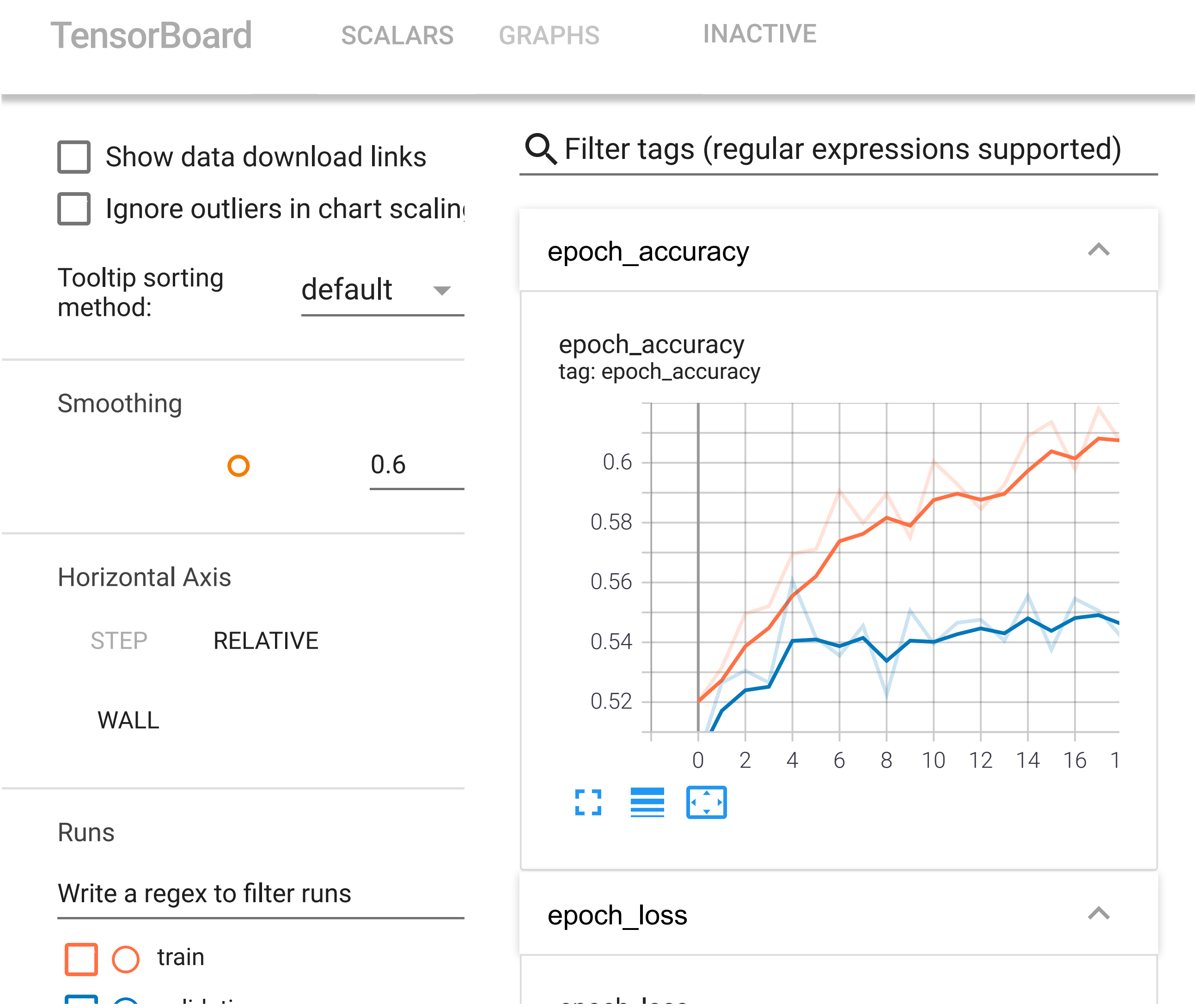
```
#compiling the model
model3.compile(optimizer='adam',loss="binary_crossentropy", metrics=["accuracy"])
```

```
#training the model, only the layers below non-trainable layers will be trained
model3.fit_generator(train_ImageGenerator, epochs = 20, validation_data=(test_Imag
```

```
Epoch 1/20
```

```
125/125 [==============================] - 159s 1s/step - loss: 0.7430 - accu
Epoch 2/20
125/125 [==============================] - 142s 1s/step - loss: 0.6945 - accu
Epoch 3/20
125/125 [==============================] - 142s 1s/step - loss: 0.6928 - accu
Epoch 4/20
125/125 [==============================] - 127s 947ms/step - loss: 0.6830 - a
Epoch 5/20
125/125 [==============================] - 129s 964ms/step - loss: 0.6835 - a
Epoch 6/20
125/125 [==============================] - 144s 1s/step - loss: 0.6776 - accu
Epoch 7/20
125/125 [==============================] - 127s 955ms/step - loss: 0.6729 - a
Epoch 8/20
125/125 [==============================] - 128s 968ms/step - loss: 0.6723 - a
Epoch 9/20
125/125 [==============================] - 142s 1s/step - loss: 0.6648 - accu
Epoch 10/20
125/125 [==============================] - 128s 958ms/step - loss: 0.6730 - a
Epoch 11/20
125/125 [==============================] - 127s 947ms/step - loss: 0.6637 - a
Epoch 12/20
125/125 [==============================] - 127s 953ms/step - loss: 0.6614 - a
Epoch 13/20
125/125 [==============================] - 127s 955ms/step - loss: 0.6659 - a
Epoch 14/20
125/125 [==============================] - 127s 962ms/step - loss: 0.6666 - a
Epoch 15/20
125/125 [==============================] - 128s 954ms/step - loss: 0.6577 - a
Epoch 16/20
125/125 [==============================] - 128s 965ms/step - loss: 0.6514 - a
Epoch 17/20
125/125 [==============================] - 129s 972ms/step - loss: 0.6572 - a
Epoch 18/20
125/125 [==============================] - 133s 959ms/step - loss: 0.6537 - a
Epoch 19/20
125/125 [==============================] - 127s 960ms/step - loss: 0.6591 - a
Epoch 20/20
125/125 [==============================] - 141s 1s/step - loss: 0.6584 - accu
<tensorflow.python.keras.callbacks.History at 0x7fe2466fc2d0>
```

```
#loading tensorboard
%tensorboard --logdir log3
```



□ Show data download links

□ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing ○ 0.6

Horizontal Axis

STEP RELATIVE

WALL

□ ○ train

```
#predicting values for test dataset
y_pred_i = model3.predict(test_ImageGenerator)
```

log3

```
y_pred_i_df = pd.DataFrame(y_pred_i).round().astype('int')
```

```
#plotting confusion matrix
cm_i = confusion_matrix(val_df['dank_or_not'], y_pred_i_df[0])
sns.heatmap(cm_i, annot=True, fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
#getting true negative, false positive, false negative and true positive values fr
tn_i, fp_i, fn_i, tp_i = cm_i.ravel()
```

```
#computing sensetivity and specificity
sensitivity_i = (tp_i/(tp_i+fn_i)).round(4)
specificity_i = (tn_i/(tn_i+fp_i)).round(4)
print('sensitivity    : ',sensitivity_i)
print('specificity    : ',specificity_i)
print('accuracy       : ',accuracy(val_df['dank_or_not'], y_pred_i_df[0]))
```

```
sensitivity    :  0.3956
specificity    :  0.6088
accuracy       :  tf.Tensor(0.5070785, shape=(), dtype=float32)
```

```
#saving the model
model3.save('inception_model.h5')
```

Applying Efficient Net model

```
#importing EfficientNetB5 model
from tensorflow.keras.applications import EfficientNetB5
```

```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./log4/
```

```
#defining EfficientNetB5 model without including top fully connected layers, to pe
efficient_model = EfficientNetB5(weights='imagenet', include_top=False, input_shap
```

```
Downloading data from https://storage.googleapis.com/keras-applications/effic
115269632/115263384 [==============================] - 1s 0us/step
```

```
#setting layers of EfficientNetB5 as non-trainable
for layer in efficient_model.layers:
  layer.trainable = False
```

```
#checking layers of EfficientNetB5
for i, layer in enumerate(efficient_model.layers):
    print(i, layer.name, layer.trainable)
```

```
511 block6h_project_conv False
512 block6h_project_bn False
513 block6h_drop False
514 block6h_add False
515 block6i_expand_conv False
516 block6i_expand_bn False
```

```
517 block6i_expand_activation False
518 block6i_dwconv False
519 block6i_bn False
520 block6i_activation False
521 block6i_se_squeeze False
522 block6i_se_reshape False
523 block6i_se_reduce False
524 block6i_se_expand False
525 block6i_se_excite False
526 block6i_project_conv False
527 block6i_project_bn False
528 block6i_drop False
529 block6i_add False
530 block7a_expand_conv False
531 block7a_expand_bn False
532 block7a_expand_activation False
533 block7a_dwconv False
534 block7a_bn False
535 block7a_activation False
536 block7a_se_squeeze False
537 block7a_se_reshape False
538 block7a_se_reduce False
539 block7a_se_expand False
540 block7a_se_excite False
541 block7a_project_conv False
542 block7a_project_bn False
543 block7b_expand_conv False
544 block7b_expand_bn False
545 block7b_expand_activation False
546 block7b_dwconv False
547 block7b_bn False
548 block7b_activation False
549 block7b_se_squeeze False
550 block7b_se_reshape False
551 block7b_se_reduce False
552 block7b_se_expand False

553 block7b_se_excite False
554 block7b_project_conv False
555 block7b_project_bn False
556 block7b_drop False
557 block7b_add False
558 block7c_expand_conv False
559 block7c_expand_bn False
560 block7c_expand_activation False
561 block7c_dwconv False
562 block7c_bn False
563 block7c_activation False
564 block7c_se_squeeze False
565 block7c_se_reshape False
566 block7c_se_reduce False
567 block7c_se_expand False
568 block7c_se_excite False
569 block7c_project_conv False
570 block7c_project_bn False
```

```
efficient_model.summary()
```

```
________________________________________________________________________________
block7b_dwconv (DepthwiseConv2D (None, 7, 7, 3072)   27648       block7b_ex
________________________________________________________________________________
```

```
block7b_bn (BatchNormalization) (None, 7, 7, 3072)   12288       block7b_dw
________________________________________________________________________________
block7b_activation (Activation) (None, 7, 7, 3072)   0           block7b_bn
________________________________________________________________________________
block7b_se_squeeze (GlobalAvera (None, 3072)         0           block7b_ac
________________________________________________________________________________
block7b_se_reshape (Reshape)    (None, 1, 1, 3072)   0           block7b_se
________________________________________________________________________________
block7b_se_reduce (Conv2D)      (None, 1, 1, 128)    393344      block7b_se
________________________________________________________________________________
block7b_se_expand (Conv2D)      (None, 1, 1, 3072)   396288      block7b_se
________________________________________________________________________________
block7b_se_excite (Multiply)    (None, 7, 7, 3072)   0           block7b_ac
                                                                 block7b_se
________________________________________________________________________________
block7b_project_conv (Conv2D)   (None, 7, 7, 512)    1572864     block7b_se
________________________________________________________________________________
block7b_project_bn (BatchNormal (None, 7, 7, 512)    2048        block7b_pr
________________________________________________________________________________
block7b_drop (Dropout)          (None, 7, 7, 512)    0           block7b_pr
________________________________________________________________________________
block7b_add (Add)               (None, 7, 7, 512)    0           block7b_dr
                                                                 block7a_pr
________________________________________________________________________________
block7c_expand_conv (Conv2D)    (None, 7, 7, 3072)   1572864     block7b_ad
________________________________________________________________________________
block7c_expand_bn (BatchNormali (None, 7, 7, 3072)   12288       block7c_ex
________________________________________________________________________________
block7c_expand_activation (Acti (None, 7, 7, 3072)   0           block7c_ex
________________________________________________________________________________
block7c_dwconv (DepthwiseConv2D (None, 7, 7, 3072)   27648       block7c_ex
________________________________________________________________________________
block7c_bn (BatchNormalization) (None, 7, 7, 3072)   12288       block7c_dw
________________________________________________________________________________
block7c_activation (Activation) (None, 7, 7, 3072)   0           block7c_bn
________________________________________________________________________________
block7c_se_squeeze (GlobalAvera (None, 3072)         0           block7c_ac
________________________________________________________________________________
block7c_se_reshape (Reshape)    (None, 1, 1, 3072)   0           block7c_se
________________________________________________________________________________
block7c_se_reduce (Conv2D)      (None, 1, 1, 128)    393344      block7c_se
________________________________________________________________________________
block7c_se_expand (Conv2D)      (None, 1, 1, 3072)   396288      block7c_se
________________________________________________________________________________
block7c_se_excite (Multiply)    (None, 7, 7, 3072)   0           block7c_ac
                                                                 block7c_se
________________________________________________________________________________
block7c_project_conv (Conv2D)   (None, 7, 7, 512)    1572864     block7c_se
________________________________________________________________________________
block7c_project_bn (BatchNormal (None, 7, 7, 512)    2048        block7c_pr
________________________________________________________________________________
block7c_drop (Dropout)          (None, 7, 7, 512)    0           block7c_pr
________________________________________________________________________________
block7c_add (Add)               (None, 7, 7, 512)    0           block7c_dr
                                                                 block7b_ad
```

```
#adding layers after non-trainable layers of EfficientNetB5
x = efficient_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu', kernel initializer='HeUniform')(x)
```

```
x = (Dense(1, activation='sigmoid'))(x)

model4 = Model(inputs=efficient_model.input, outputs=x)


#tensorboard callback
tbCallBack4 = tf.keras.callbacks.TensorBoard(log_dir='./log4', histogram_freq=0,
                         write_graph=True,
                         write_grads=True,
                         write_images=True)
callback_list4 = [tbCallBack4]
```

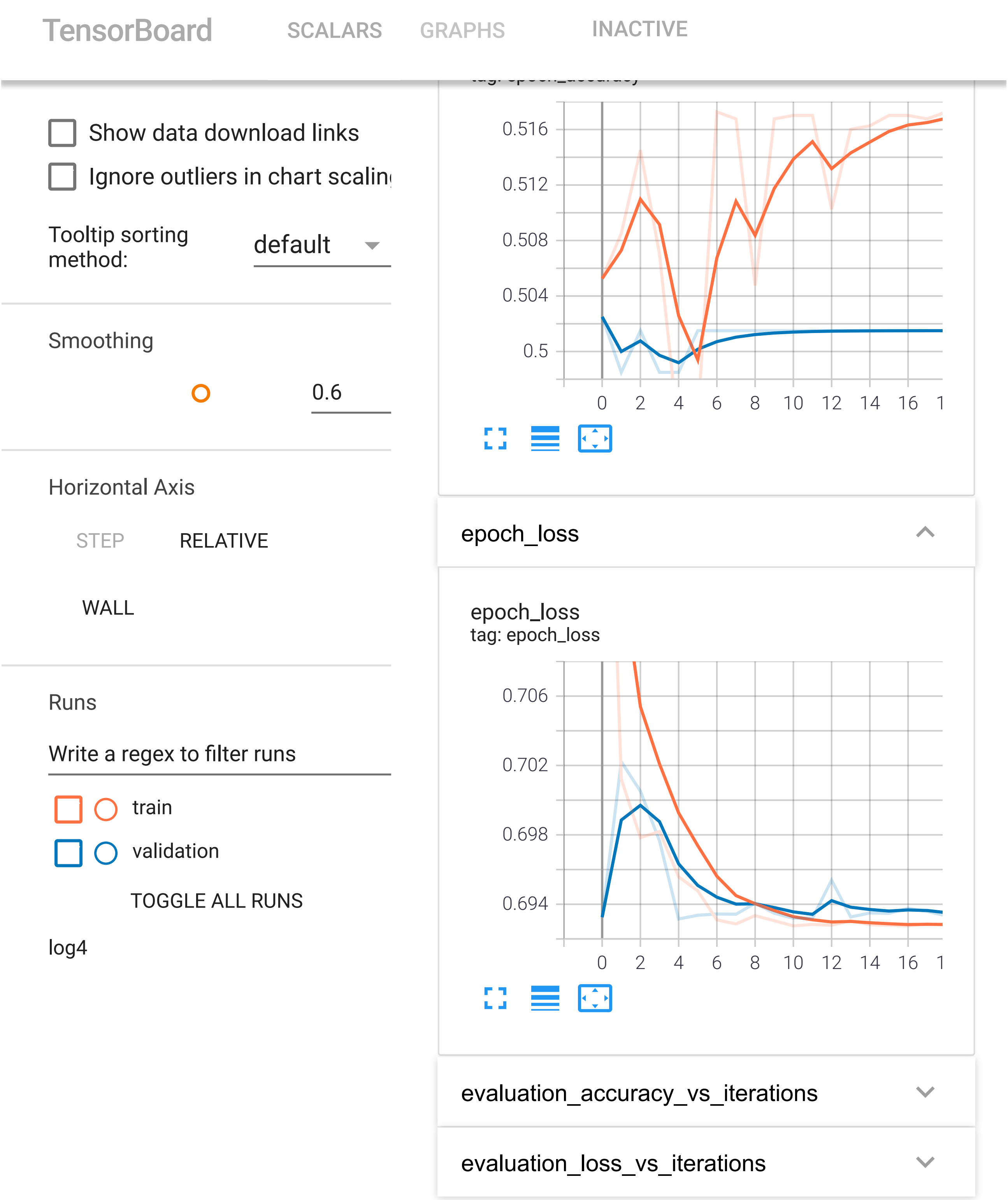
```
    WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `T
```

```
#compiling the model
model4.compile(optimizer='adam',loss="binary_crossentropy", metrics=["accuracy"])


#training the model, only the layers below non-trainable layers will be trained
model4.fit_generator(train_ImageGenerator, epochs = 20, validation_data=(test_Imag
```

```
    Epoch 1/20
    125/125 [==============================] - 233s 1s/step - loss: 0.7333 - accu
    Epoch 2/20
    125/125 [==============================] - 132s 996ms/step - loss: 0.7012 - a
    Epoch 3/20
    125/125 [==============================] - 146s 1s/step - loss: 0.6978 - accu
    Epoch 4/20
    125/125 [==============================] - 145s 1s/step - loss: 0.6982 - accu
    Epoch 5/20
    125/125 [==============================] - 132s 999ms/step - loss: 0.6956 - a
    Epoch 6/20
    125/125 [==============================] - 145s 1s/step - loss: 0.6948 - accu
    Epoch 7/20
    125/125 [==============================] - 131s 983ms/step - loss: 0.6931 - a
    Epoch 8/20
    125/125 [==============================] - 129s 966ms/step - loss: 0.6929 - a
    Epoch 9/20
    125/125 [==============================] - 130s 968ms/step - loss: 0.6933 - a
    Epoch 10/20
    125/125 [==============================] - 130s 983ms/step - loss: 0.6930 - a
    Epoch 11/20
    125/125 [==============================] - 144s 1s/step - loss: 0.6928 - accu
    Epoch 12/20
    125/125 [==============================] - 131s 976ms/step - loss: 0.6928 - a
    Epoch 13/20
    125/125 [==============================] - 135s 985ms/step - loss: 0.6928 - a
    Epoch 14/20
    125/125 [==============================] - 143s 1s/step - loss: 0.6931 - accu
    Epoch 15/20
    125/125 [==============================] - 131s 981ms/step - loss: 0.6928 - a
    Epoch 16/20
    125/125 [==============================] - 145s 1s/step - loss: 0.6928 - accu
    Epoch 17/20
    125/125 [==============================] - 145s 1s/step - loss: 0.6928 - accu
    Epoch 18/20
    125/125 [==============================] - 129s 964ms/step - loss: 0.6929 - a
    Epoch 19/20
```

```
125/125 [==============================] - 150s 1s/step - loss: 0.6928 - accu
Epoch 20/20
125/125 [==============================] - 130s 981ms/step - loss: 0.6926 - a
<tensorflow.python.keras.callbacks.History at 0x7fe25dc02210>
```

```
#loading the tensorboard
%tensorboard --logdir log4
```



□ 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

□ ○ train

□ ○ validation

TOGGLE ALL RUNS

log4

tag: epoch_accuracy

epoch_loss

epoch_loss
tag: epoch_loss

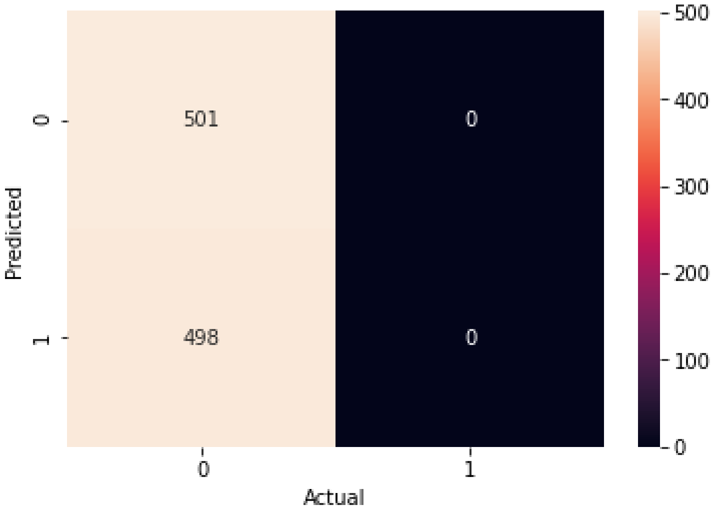
evaluation_accuracy_vs_iterations

evaluation_loss_vs_iterations

```
#predicting the test data
y_pred_e = model4.predict(test_ImageGenerator)
```

```
y_pred_e_df = pd.DataFrame(y_pred_e).round().astype('int')
```

```
#plotting confusion matrix
cm_e = confusion_matrix(val_df['dank_or_not'], y_pred_e_df[0])
sns.heatmap(cm_e, annot=True, fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



```
#getting true negative, false positive, false negative and true positive values fr
tn_e, fp_e, fn_e, tp_e = cm_e.ravel()
```

```
#computing sensetivity and specificity
sensitivity_e = (tp_e/(tp_e+fn_e)).round(4)
specificity_e = (tn_e/(tn_e+fp_e)).round(4)
print('sensitivity    : ',sensitivity_e)
print('specificity    : ',specificity_e)
print('accuracy       : ',accuracy(val_df['dank_or_not'], y_pred_e_df[0]))

    sensitivity    :  0.0
    specificity    :  1.0
    accuracy       :  tf.Tensor(0.5063814, shape=(), dtype=float32)
```

```
#saving the model
model4.save('efficientnet_model.h5')
```

Applying Resnet50

```
#importing ResNet50 model
from tensorflow.keras.applications.resnet50 import ResNet50
```

```
%load_ext tensorboard
# Clear any logs from previous runs
```

```
!rm -rf ./log5/

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

```
#defining ResNet50 model without including top fully connected layers, to perform
resnet_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 2

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applica
    94773248/94765736 [==============================] - 1s 0us/step
```

```
#setting layers of ResNet50 as non-trainable
for layer in resnet_model.layers:
  layer.trainable = False
```

```
#checking layers of ResNet50
for i, layer in enumerate(resnet_model.layers):
    print(i, layer.name, layer.trainable)
    114 conv4_block4_1_bn False
    115 conv4_block4_1_relu False
    116 conv4_block4_2_conv False
    117 conv4_block4_2_bn False
    118 conv4_block4_2_relu False
    119 conv4_block4_3_conv False
    120 conv4_block4_3_bn False
    121 conv4_block4_add False
    122 conv4_block4_out False
    123 conv4_block5_1_conv False
    124 conv4_block5_1_bn False
    125 conv4_block5_1_relu False
    126 conv4_block5_2_conv False
    127 conv4_block5_2_bn False
    128 conv4_block5_2_relu False
    129 conv4_block5_3_conv False
    130 conv4_block5_3_bn False
    131 conv4_block5_add False
    132 conv4_block5_out False
    133 conv4_block6_1_conv False
    134 conv4_block6_1_bn False
    135 conv4_block6_1_relu False
    136 conv4_block6_2_conv False
    137 conv4_block6_2_bn False
    138 conv4_block6_2_relu False
    139 conv4_block6_3_conv False
    140 conv4_block6_3_bn False
    141 conv4_block6_add False
    142 conv4_block6_out False
    143 conv5_block1_1_conv False
    144 conv5_block1_1_bn False
    145 conv5_block1_1_relu False
    146 conv5_block1_2_conv False
    147 conv5_block1_2_bn False
    148 conv5_block1_2_relu False
    149 conv5_block1_0_conv False
    150 conv5_block1_3_conv False
    151 conv5_block1_0_bn False
    152 conv5_block1_3_bn False
```

```
152 conv5_block1_3_bn False
153 conv5_block1_add False
154 conv5_block1_out False
155 conv5_block2_1_conv False
156 conv5_block2_1_bn False
157 conv5_block2_1_relu False
158 conv5_block2_2_conv False

159 conv5_block2_2_bn False
160 conv5_block2_2_relu False
161 conv5_block2_3_conv False
162 conv5_block2_3_bn False
163 conv5_block2_add False
164 conv5_block2_out False
165 conv5_block3_1_conv False
166 conv5_block3_1_bn False
167 conv5_block3_1_relu False
168 conv5_block3_2_conv False
169 conv5_block3_2_bn False
170 conv5_block3_2_relu False
171 conv5_block3_3_conv False
172 conv5_block3_3_bn False
173 conv5_block3_add False
```

```
resnet_model.summary()
```

```
conv5_block1_2_relu (Activation (None, 7, 7, 512)    0           conv5_bloc

conv5_block1_0_conv (Conv2D)    (None, 7, 7, 2048)   2099200     conv4_bloc

conv5_block1_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_bloc

conv5_block1_0_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_bloc

conv5_block1_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_bloc

conv5_block1_add (Add)          (None, 7, 7, 2048)   0           conv5_bloc
                                                                 conv5_bloc

conv5_block1_out (Activation)   (None, 7, 7, 2048)   0           conv5_bloc

conv5_block2_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_bloc

conv5_block2_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_bloc

conv5_block2_1_relu (Activation (None, 7, 7, 512)    0           conv5_bloc

conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_bloc

conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_bloc

conv5_block2_2_relu (Activation (None, 7, 7, 512)    0           conv5_bloc

conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_bloc


conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_bloc

conv5_block2_add (Add)          (None, 7, 7, 2048)   0           conv5_bloc
                                                                 conv5_bloc

conv5_block2_out (Activation)   (None, 7, 7, 2048)   0           conv5_bloc
```

```
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_bloc
________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_bloc
________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0           conv5_bloc
________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_bloc
________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_bloc
________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0           conv5_bloc
________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_bloc
________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_bloc
________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_bloc
                                                                 conv5_bloc
________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_bloc
================================================================================
Total params: 23,587,712
```

```
#adding layers after non-trainable layers of ResNet50
x = resnet_model.output

x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu', kernel_initializer='HeUniform')(x)
x = (Dense(1, activation='sigmoid'))(x)

model5 = Model(inputs=resnet_model.input, outputs=x)


#tensorboard callback
tbCallBack5 = tf.keras.callbacks.TensorBoard(log_dir='./log5', histogram_freq=0,
                         write_graph=True,
                         write_grads=True,
                         write_images=True)
callback_list5 = [tbCallBack5]
```

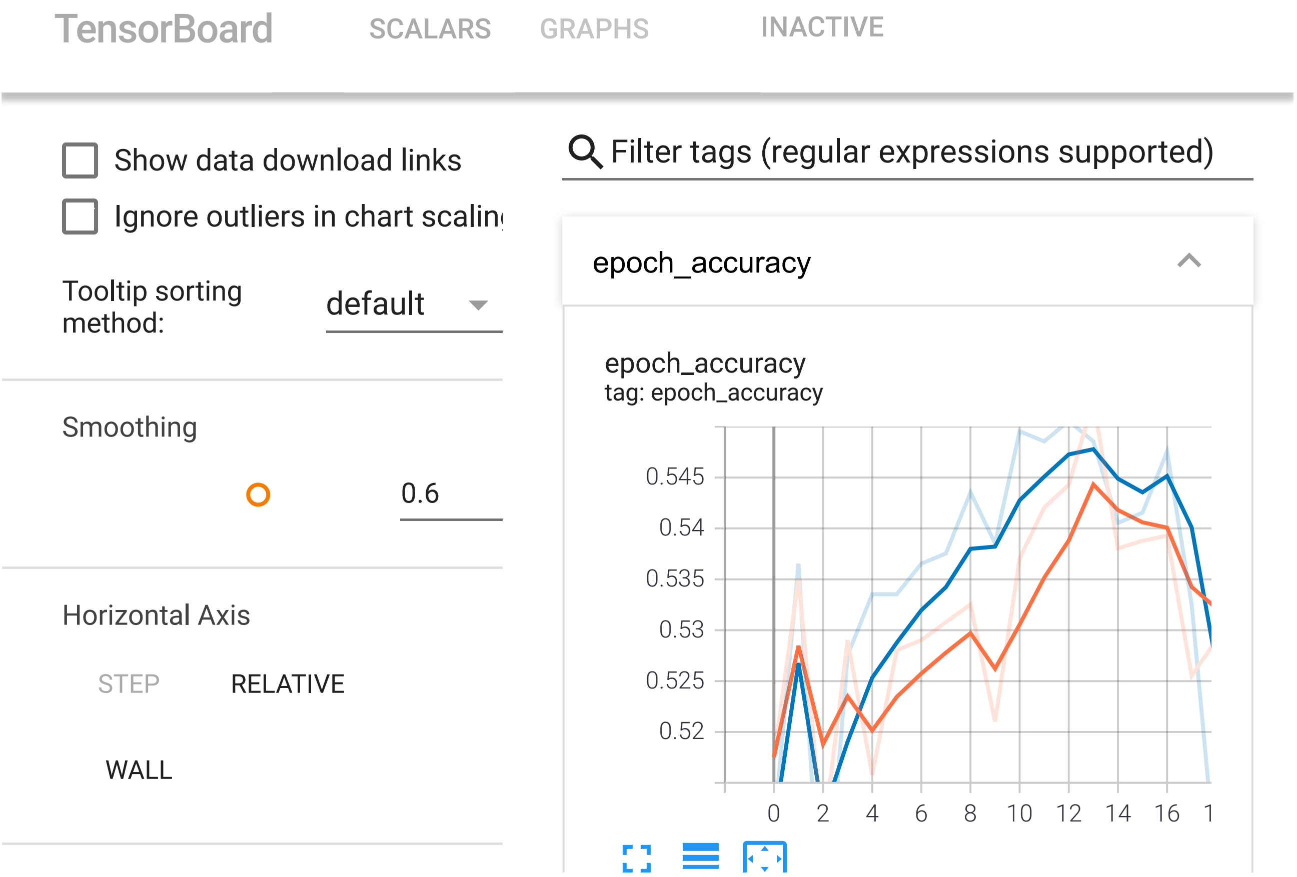
```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `T
```

```
#compiling the model
model5.compile(optimizer='adam',loss="binary_crossentropy", metrics=["accuracy"])


#training the model, only the layers below non-trainable layers will be trained
model5.fit_generator(train_ImageGenerator, epochs = 20, validation_data=(test_Imag
```

```
Epoch 1/20
125/125 [==============================] - 135s 992ms/step - loss: 0.7400 - a
Epoch 2/20
125/125 [==============================] - 128s 973ms/step - loss: 0.7046 - a
Epoch 3/20
125/125 [==============================] - 128s 976ms/step - loss: 0.7197 - a
Epoch 4/20
```

```
125/125 [==============================] - 129s 962ms/step - loss: 0.7088 - a
Epoch 5/20
125/125 [==============================] - 128s 969ms/step - loss: 0.6980 - a
Epoch 6/20
125/125 [==============================] - 129s 963ms/step - loss: 0.6938 - a
Epoch 7/20
125/125 [==============================] - 129s 960ms/step - loss: 0.6921 - a
Epoch 8/20
125/125 [==============================] - 129s 978ms/step - loss: 0.6924 - a
Epoch 9/20
125/125 [==============================] - 146s 1s/step - loss: 0.6897 - accu
Epoch 10/20
125/125 [==============================] - 132s 984ms/step - loss: 0.6943 - a
Epoch 11/20
125/125 [==============================] - 135s 969ms/step - loss: 0.6884 - a
Epoch 12/20
125/125 [==============================] - 128s 966ms/step - loss: 0.6883 - a
Epoch 13/20
125/125 [==============================] - 128s 953ms/step - loss: 0.6881 - a
Epoch 14/20
125/125 [==============================] - 129s 970ms/step - loss: 0.6859 - a
Epoch 15/20
125/125 [==============================] - 129s 967ms/step - loss: 0.6894 - a
Epoch 16/20
125/125 [==============================] - 130s 973ms/step - loss: 0.6872 - a
Epoch 17/20
125/125 [==============================] - 131s 972ms/step - loss: 0.6883 - a
Epoch 18/20
125/125 [==============================] - 145s 1s/step - loss: 0.6891 - accu
Epoch 19/20
125/125 [==============================] - 132s 989ms/step - loss: 0.6889 - a
Epoch 20/20
125/125 [==============================] - 133s 1s/step - loss: 0.6886 - accu
<tensorflow.python.keras.callbacks.History at 0x7fe30cec8090>
```

```
#loading the tensorboard
%tensorboard --logdir log5
```



□ Show data download links

□ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

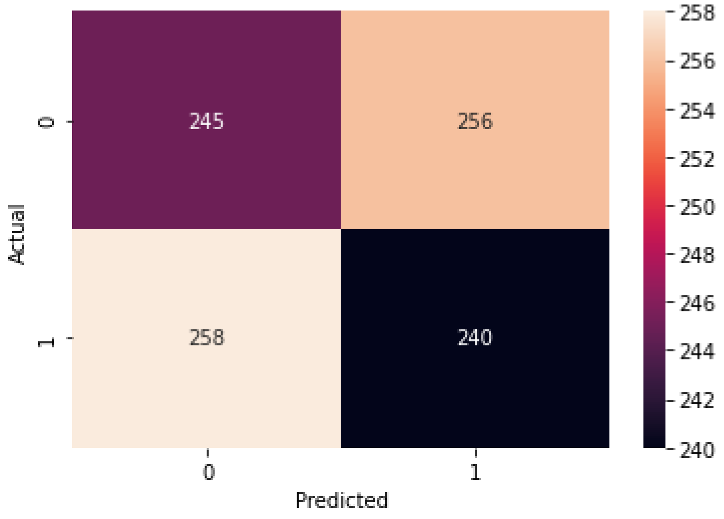
STEP RELATIVE WALL

Filter tags (regular expressions supported)

epoch_accuracy

epoch_accuracy
tag: epoch_accuracy

```
#predicting the test data
y_pred_r = model5.predict(test_ImageGenerator)

y_pred_r_df = pd.DataFrame(y_pred_r).round().astype('int')

#plotting confusion matrix
cm_r = confusion_matrix(val_df['dank_or_not'], y_pred_r_df[0])
sns.heatmap(cm_r, annot=True, fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



```
#getting true negative, false positive, false negative and true positive values fr
```

```
tn_r, fp_r, fn_r, tp_r = cm_r.ravel()


#computing sensitivity and specificity
sensitivity_r = (tp_r/(tp_r+fn_r)).round(4)
specificity_r = (tn_r/(tn_r+fp_r)).round(4)
print('sensitivity    : ',sensitivity_r)
print('specificity    : ',specificity_r)
print('accuracy       : ',accuracy(val_df['dank_or_not'], y_pred_r_df[0]))
```

```
    sensitivity    :  0.4819
    specificity    :  0.489
    accuracy       :  tf.Tensor(0.50158495, shape=(), dtype=float32)
```

```
#saving the model
model5.save('resnet_model.h5')
```

Comparing all models

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model Name", "Sensitivity", "Specificity"]
x.add_row(["Baseline", 0.50, 0.51])
x.add_row(["VGG16", 0.4719, 0.5269])
x.add_row(["InceptionV3", 0.3956, 0.6088])
x.add_row(["EfficientNetB5", 0.0, 1.0])
x.add_row(["ResNet50", 0.4819, 0.489])


print(x)
```

```
    +----------------+-------------+-------------+
    |   Model Name   | Sensitivity | Specificity |
    +----------------+-------------+-------------+
    |    Baseline    |     0.5     |     0.51    |
    |     VGG16      |    0.4719   |    0.5269   |
    |  InceptionV3   |    0.3956   |    0.6088   |
    | EfficientNetB5 |     0.0     |     1.0     |
    |    ResNet50    |    0.4819   |    0.489    |
    +----------------+-------------+-------------+
```

From the table we can see that ResNet50 and VGG16 are best, InceptionV3 is also performing well and EfficientNetB5 is worst.

✓ 0s completed at 12:32 PM ● ×