```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

import cv2
from prettytable import PrettyTable
from sklearn.model_selection import train_test_split
import random
from skimage.util import view_as_windows
from matplotlib.pyplot import imread, imsave

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.callbacks import ModelCheckpoint,LearningRateScheduler, EarlyStopping, TensorBo
import tensorflow as tf
import datetime

from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
from scipy.sparse import csr_matrix

from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
import keras
from keras.applications import *


from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50

from keras_preprocessing.image import ImageDataGenerator
from keras.models import Model
from keras.layers import Activation, Conv2D, Input, Embedding, Reshape, MaxPooling2D,MaxPo
from keras import optimizers
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications import EfficientNetB0



from efficientnet.tfkeras import preprocess_input
from efficientnet import model
from keras.models import Model, load_model
from keras.callbacks import EarlyStopping, ModelCheckpoint, LearningRateScheduler,TensorBo
from keras import optimizers
from keras.layers import BatchNormalization
from keras.layers import BatchNormalization


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

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.r
```

```
#Initialise all the director paths and fetch the filenames from this directory.
#The dataset consists of images belonging to Van Gogh and other artists. The images are se
#The train directory and test directory each have two folders - vg and nvg.
#vg folder has images belonging to Van Gogh
#nvg folder has images belonginh to Non Van Gogh

rootdir = '/content/drive/MyDrive/Colab Notebooks/29. Identification of Van Gogh paintings

#Train data
traindir = rootdir + '/train'
traindir_vg = traindir + '/vg'
traindir_nvg = traindir + '/nvg'

#Test data
testdir = rootdir + '/test'
testdir_vg = testdir + '/vg'
testdir_nvg = testdir + '/nvg'

train_vg_data = os.listdir(traindir_vg)
train_nvg_data = os.listdir(traindir_nvg)

test_vg_data = os.listdir(testdir_vg)
test_nvg_data = os.listdir(testdir_nvg)

dir_patch_tr=traindir+'/patches'
dir_patch_te=testdir+'/patches'
```

Resnet

```
resnet50model = ResNet50(include_top = False, weights = 'imagenet')

     Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/re
     94773248/94765736 [==============================] - 1s 0us/step
     94781440/94765736 [==============================] - 1s 0us/step
```

```
resnet50model.summary()

     Model: "resnet50"
     ____________________________________________________________________________________
     Layer (type)                    Output Shape         Param #     Connected to
     ====================================================================================
     input_1 (InputLayer)            [(None, None, None,  0
     ____________________________________________________________________________________
```

```
conv1_pad (ZeroPadding2D)       (None, None, None, 3 0          input_1[0][0]

conv1_conv (Conv2D)             (None, None, None, 6 9472       conv1_pad[0][0]

conv1_bn (BatchNormalization)   (None, None, None, 6 256        conv1_conv[0][0]

conv1_relu (Activation)         (None, None, None, 6 0          conv1_bn[0][0]

pool1_pad (ZeroPadding2D)       (None, None, None, 6 0          conv1_relu[0][0]

pool1_pool (MaxPooling2D)       (None, None, None, 6 0          pool1_pad[0][0]

conv2_block1_1_conv (Conv2D)    (None, None, None, 6 4160       pool1_pool[0][0]

conv2_block1_1_bn (BatchNormali (None, None, None, 6 256        conv2_block1_1_co

conv2_block1_1_relu (Activation (None, None, None, 6 0          conv2_block1_1_bn

conv2_block1_2_conv (Conv2D)    (None, None, None, 6 36928      conv2_block1_1_re

conv2_block1_2_bn (BatchNormali (None, None, None, 6 256        conv2_block1_2_co

conv2_block1_2_relu (Activation (None, None, None, 6 0          conv2_block1_2_bn

conv2_block1_0_conv (Conv2D)    (None, None, None, 2 16640      pool1_pool[0][0]

conv2_block1_3_conv (Conv2D)    (None, None, None, 2 16640      conv2_block1_2_re

conv2_block1_0_bn (BatchNormali (None, None, None, 2 1024       conv2_block1_0_co

conv2_block1_3_bn (BatchNormali (None, None, None, 2 1024       conv2_block1_3_co

conv2_block1_add (Add)          (None, None, None, 2 0          conv2_block1_0_bn
                                                                conv2_block1_3_bn

conv2_block1_out (Activation)   (None, None, None, 2 0          conv2_block1_add[

conv2_block2_1_conv (Conv2D)    (None, None, None, 6 16448      conv2_block1_out[

conv2_block2_1_bn (BatchNormali (None, None, None, 6 256        conv2_block2_1_co

conv2_block2_1_relu (Activation (None, None, None, 6 0          conv2_block2_1_bn

conv2_block2_2_conv (Conv2D)    (None, None, None, 6 36928      conv2_block2_1_re

conv2_block2_2_bn (BatchNormali (None, None, None, 6 256        conv2_block2_2_co

conv2_block2_2_relu (Activation (None, None, None, 6 0          conv2_block2_2_bn

conv2_block2_3_conv (Conv2D)    (None, None, None, 2 16640      conv2_block2_2_re

conv2_block2_3_bn (BatchNormali (None, None, None, 2 1024       conv2_block2_3_co
```

```
'''This function is used to generate patches for a particular image.
The number of patches and patch_size are passed as function parameters.
The function generates the number of patches as the given parameter'''
def get_patches_for_img(img_path, patch_size, no_of_patches):

    patches = []
    #read the image at the given path
    img = cv2.imread(img_path, 1)
    image_height = img.shape[0]
    image_width = img.shape[1]

    #Subtract patch_size from image's height and width to avoid out of bounds error
    range_x = image_height - patch_size
    range_y = image_width - patch_size

    #Generate patches for each image. The number of patches are passed as parameter.
    for i in range(no_of_patches):

        #Generate patch from random area of the image
        x = np.random.randint(low = 0, high = range_x)
        y = np.random.randint(low = 0, high = range_y)

        #The patch is calculated by adding the patch_size to both x and y co-ordinates
        patch = img[x : x+patch_size, y : y+patch_size, :]
        patches.append(patch)

    return patches

'''This function creates a feature array for a patch.
It uses the VGG 19 to pre-process and predict the feature array for the patch. '''
def get_patch_feature_resnet50(patch):

    patch_input = np.expand_dims(patch, axis = 0)             #add an extra dimension for
    patch_preprocessed_input = preprocess_input(patch_input)

    p_feature = resnet50model.predict(patch_preprocessed_input)
    p_feature = p_feature.reshape(100352)

    return p_feature


patch_size = 224                            #patch_size for resnet model
X_train = []
Y_train = []

'''
Generate patches and create x,y array
'''
def get_x_y_arr(filename_list,dir_name,y_value,no_of_patches):

  x = []
  y = []

  for file_name in filename_list:

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)

    for patch in patches:
        patch_feature = get_patch_feature_resnet50(patch)
        x.append(patch_feature)
        x.append(patch_feature)
        y.append(y_value)

  return  x,y
```

```
#30 patches are created for VG data and 20 patches are created for NVG data. This is done
X_train_vg, Y_train_vg = get_x_y_arr(train_vg_data,traindir_vg,1,30)
X_train_nvg, Y_train_nvg = get_x_y_arr(train_nvg_data,traindir_nvg,0,20)

print("The number of rows in VG-Data set are: ",len(X_train_vg))
print("The number of rows in NVG-Data set are: ",len(X_train_nvg))
```

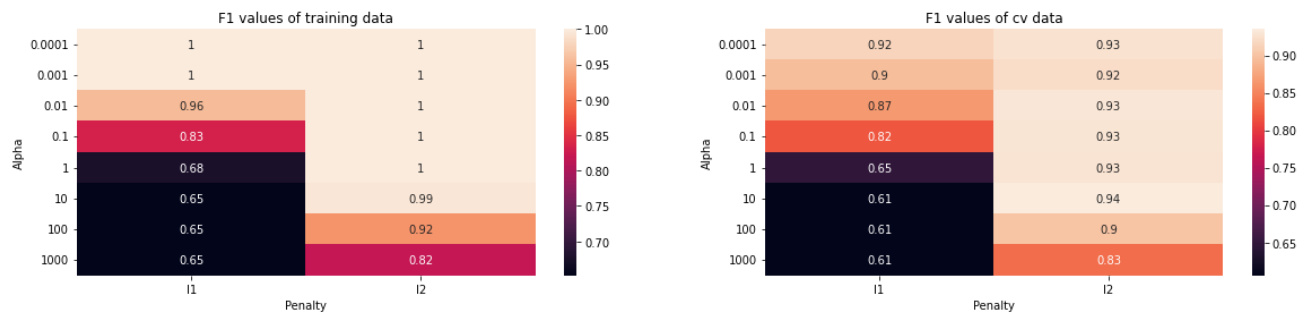
```
The number of rows in VG-Data set are:  2970
The number of rows in NVG-Data set are:  3300
```

```
X_train = []
X_train.extend(X_train_vg)
X_train.extend(X_train_nvg)

Y_train = []
Y_train.extend(Y_train_vg)
Y_train.extend(Y_train_nvg)


X_train, Y_train = shuffle(X_train, Y_train)
X_train = csr_matrix(X_train)

X_tr, X_cv, Y_tr, Y_cv = train_test_split(X_train, Y_train, test_size = 0.2, random_state
print("Shape of training data, CV data is :", X_tr.shape, X_cv.shape)
```

```
Shape of training data, CV data is : (5016, 100352) (1254, 100352)
```

SVM

```
#Hyperparameter tuning
alpha = [10**i for i in range(-4, 4)]
penalty = ['l1', 'l2']

f1_score_tr = []
f1_score_cv = []

for i in alpha:
    for j in penalty:

        clf = SGDClassifier(alpha = i, penalty = j, loss = 'hinge')
        clf.fit(X_tr, Y_tr)

        pred_tr = clf.predict(X_tr)
        pred_cv = clf.predict(X_cv)

        f1_score_tr.append(f1_score(Y_tr, pred_tr))
        f1_score_cv.append(f1_score(Y_cv, pred_cv))



plt.figure(figsize = (20,4))
plt.subplot(1, 2, 1)
f1_score_tr = np.array(f1_score_tr)
f1_score_tr = f1_score_tr.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_tr, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of training data')

plt.subplot(1, 2, 2)
f1_score_cv = np.array(f1_score_cv)
f1_score_cv = f1_score_cv.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_cv, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of cv data')
plt.show()
```



```
#Training with best parameters
clf = SGDClassifier(alpha = 1, penalty = 'l2', loss = 'hinge')
clf.fit(X_tr, Y_tr)

patch_size = 224
no_of_patches = 20
y = []
pred_vals = []

def get_x_y_arr_probabs(filename_list,dir_name,y_value):

  for file_name in filename_list:
    patch_pred = []

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)
```

```
      for patch in patches:
          patch_feature = get_patch_feature_resnet50(patch)

          pred_proba = clf.decision_function([patch_feature])                    #predict pr
          patch_pred.append(pred_proba)


      pred_vals.append(patch_pred)
      y.append(y_value)

    return  pred_vals,y

  get_x_y_arr_probabs(test_vg_data,testdir_vg,1)
  a,b = get_x_y_arr_probabs(test_nvg_data,testdir_nvg,0)


  #Fusion methods

  def agg_pred_far(pred):

    arr_pos = []
    arr_neg = []

    for predItem in pred:
      if(predItem >= 0):
        arr_pos.append(predItem)
      else:
        arr_neg.append(predItem)

    #arr_pos = pred[pred >= 0]
    max_pos = np.max(arr_pos) if(len(arr_pos) > 0) else 0

    #arr_neg = pred[pred <= 0]
    max_neg = np.abs(np.min(arr_neg)) if(len(arr_neg) > 0) else 0

    cl = 1 if(max_pos > max_neg) else 0

    return cl

  def agg_pred_mean(pred):
    arr_pos = []
    arr_neg = []

    for predItem in pred:
      if(predItem >= 0):
        arr_pos.append(predItem)
      else:
        arr_neg.append(predItem)


    #arr_pos = pred[pred >= 0]
    avg_pos = np.mean(arr_pos) if(len(arr_pos) > 0) else 0

    #arr_neg = pred[pred <= 0]
    avg_neg = np.abs(np.mean(arr_neg)) if(len(arr_neg) > 0) else 0
    cl = 1 if(avg_pos > avg_neg) else 0

    return cl

  def agg_pred_median(pred):

    arr_pos = []
    arr_neg = []

    for predItem in pred:
      if(predItem >= 0):
        arr_pos.append(predItem)
      else:
        arr_neg.append(predItem)
    #arr_pos = pred[pred >= 0]
    avg_pos = np.median(arr_pos) if(len(arr_pos) > 0) else 0

    #arr_neg = pred[pred <= 0]
    avg_neg = np.abs(np.median(arr_neg)) if(len(arr_neg) > 0) else 0

    cl = 1 if(avg_pos > avg_neg) else 0

    return cl


  y_pred_mean = []
  y_pred_median = []
  y_pred_far = []

  for item in pred_vals:
    y_pred_median.append(agg_pred_median(item))
    y_pred_mean.append(agg_pred_mean(item))
    y_pred_far.append(agg_pred_far(item))



  print("F1 scores with different Fusion methods")
  print("="*200)
  print("F1 score for test data - Median is ", f1_score(y, y_pred_median))
  print("F1 score for test data - Mean is", f1_score(y, y_pred_mean))
  print("F1 score for test data - Far is", f1_score(y, y_pred_far))
```

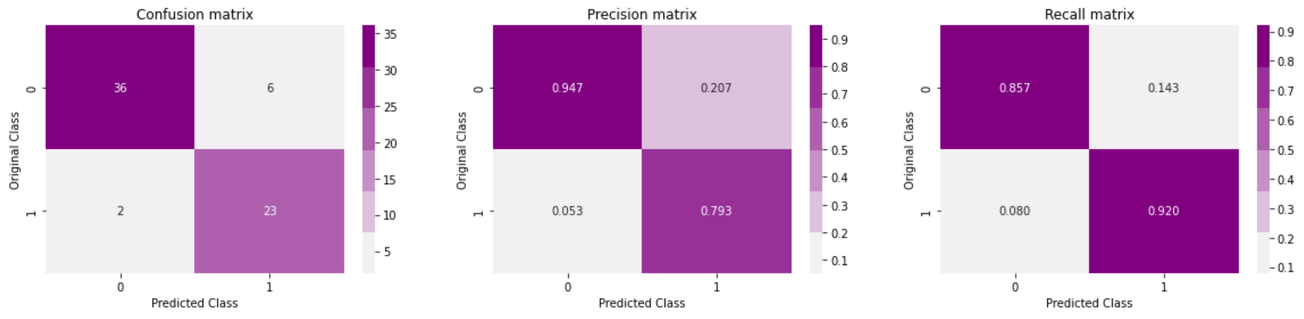
```
F1 scores with different Fusion methods
==============================================================================================
F1 score for test data - Median is  0.851851851851852
F1 score for test data - Mean is 0.851851851851852
F1 score for test data - Far is 0.8727272727272728
```

```
  def plot_confusion_matrix(test_y, predict_y):
      C = confusion_matrix(test_y, predict_y)
      A = (((C.T)/(C.sum(axis=1))).T)
      B = (C/C.sum(axis=0))
      plt.figure(figsize=(20,4))
      labels = [0,1]
```

```
    # representing A in heatmap format
    cmap = sns.light_palette("purple")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, fmt="g", xticklabels=labels, yticklabels=labels,cmap=cmap)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, fmt=".3f", xticklabels=labels, yticklabels=labels,cmap=cmap
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, fmt=".3f", xticklabels=labels, yticklabels=labels,cmap=cmap
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")
    plt.show()
```

```
f1_test_vgg19_svm = f1_score(y, y_pred_median)

print("F1 score for test data is", f1_test_vgg19_svm)
plot_confusion_matrix(y, y_pred_median)
```

```
    F1 score for test data is 0.851851851851852
```



ResNet with custom layers

```
resnet50model = ResNet50(include_top = False, weights = 'imagenet',input_shape=(224,224,3)
```

```
resnet50model.summary()
```

```
conv4_block2_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block1_out[
conv4_block2_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2_1_co
conv4_block2_1_relu (Activation (None, 14, 14, 256)  0           conv4_block2_1_bn
conv4_block2_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block2_1_re
conv4_block2_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2_2_co
conv4_block2_2_relu (Activation (None, 14, 14, 256)  0           conv4_block2_2_bn
conv4_block2_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block2_2_re
conv4_block2_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block2_3_co
conv4_block2_add (Add)          (None, 14, 14, 1024) 0           conv4_block1_out[
                                                                 conv4_block2_3_bn
conv4_block2_out (Activation)   (None, 14, 14, 1024) 0           conv4_block2_add[
conv4_block3_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block2_out[
conv4_block3_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3_1_co
conv4_block3_1_relu (Activation (None, 14, 14, 256)  0           conv4_block3_1_bn
conv4_block3_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block3_1_re
conv4_block3_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3_2_co
conv4_block3_2_relu (Activation (None, 14, 14, 256)  0           conv4_block3_2_bn
conv4_block3_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block3_2_re
conv4_block3_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block3_3_co
conv4_block3_add (Add)          (None, 14, 14, 1024) 0           conv4_block2_out[
                                                                 conv4_block3_3_bn
conv4_block3_out (Activation)   (None, 14, 14, 1024) 0           conv4_block3_add[
conv4_block4_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block3_out[
conv4_block4_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block4_1_co
conv4_block4_1_relu (Activation (None, 14, 14, 256)  0           conv4_block4_1_bn
conv4_block4_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block4_1_re
conv4_block4_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block4_2_co
conv4_block4_2_relu (Activation (None, 14, 14, 256)  0           conv4_block4_2_bn
conv4_block4_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block4_2_re
conv4_block4_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block4_3_co
```

```
#Create Image generators with data augmentation
```

```
batch_size = 16
traindir1 = traindir + '/'
testdir1 = testdir + '/'
img_height = 224
img_width = 224
# prepare data augmentation configuration
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        traindir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        testdir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='binary')
```

```
     Found 264 images belonging to 2 classes.
     Found 67 images belonging to 2 classes.
```

```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=validation_generator.n//validation_generator.batch_size

print('Step size for training data = ',STEP_SIZE_TRAIN)
print('Step size for test data = ',STEP_SIZE_VALID)
```

```
     Step size for training data =  16
     Step size for test data =  4
```

```
#Freeze the layers of the Resnet model

for layers in (resnet50model.layers)[:25]:
    print(layers)
    layers.trainable = False



res_output = resnet50model.get_layer('conv5_block3_3_conv').output

#Create a Convolutional block and fully connected layers and add it to the bottom of the R

conv_21=Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(res_outp
mx_1= MaxPool2D(pool_size=(2,2),strides=(2,2))(conv_21)
top_fc1 = Flatten()(mx_1)
top_fc2 = Dropout(0.5)(top_fc1)
```

```
    top_dense = Dense(1000,activation='relu')(top_fc2)
    top_preds = Dense(1, activation="sigmoid")(top_dense)
    resnet50model_custom = Model(resnet50model.input,top_preds)

    resnet50model_custom.summary()
```

```
conv3_block3_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block3_1_co
conv3_block3_1_relu (Activation (None, 28, 28, 128)  0           conv3_block3_1_bn
conv3_block3_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block3_1_re
conv3_block3_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block3_2_co
conv3_block3_2_relu (Activation (None, 28, 28, 128)  0           conv3_block3_2_bn
conv3_block3_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block3_2_re
conv3_block3_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block3_3_co
conv3_block3_add (Add)          (None, 28, 28, 512)  0           conv3_block2_out[
                                                                 conv3_block3_3_bn
conv3_block3_out (Activation)   (None, 28, 28, 512)  0           conv3_block3_add[
conv3_block4_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block3_out[
conv3_block4_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block4_1_co
conv3_block4_1_relu (Activation (None, 28, 28, 128)  0           conv3_block4_1_bn
conv3_block4_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block4_1_re
conv3_block4_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block4_2_co
conv3_block4_2_relu (Activation (None, 28, 28, 128)  0           conv3_block4_2_bn
conv3_block4_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block4_2_re
conv3_block4_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block4_3_co
conv3_block4_add (Add)          (None, 28, 28, 512)  0           conv3_block3_out[
                                                                 conv3_block4_3_bn
conv3_block4_out (Activation)   (None, 28, 28, 512)  0           conv3_block4_add[
conv4_block1_1_conv (Conv2D)    (None, 14, 14, 256)  131328      conv3_block4_out[
conv4_block1_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block1_1_co
conv4_block1_1_relu (Activation (None, 14, 14, 256)  0           conv4_block1_1_bn
conv4_block1_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block1_1_re
conv4_block1_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block1_2_co
conv4_block1_2_relu (Activation (None, 14, 14, 256)  0           conv4_block1_2_bn
conv4_block1_0_conv (Conv2D)    (None, 14, 14, 1024) 525312      conv3_block4_out[

conv4_block1_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block1_2_re

conv4_block1_0_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block1_0_co
```

```
resnet50model_custom.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.SGD


#Fit the model

earlyStop = EarlyStopping(monitor='val_acc', patience=10, restore_best_weights=True, verbo
cp_callback  = ModelCheckpoint(filepath=rootdir+'resnet50_modified.hdf5', monitor='val_acc
TB = TensorBoard(log_dir=rootdir+'/logs/resnet50_modified/'+datetime.datetime.now().strfti
                                          embeddings_freq=0, embeddings_layer_names=None,
                                          embeddings_metadata=None, embeddings_data=None,
                                          update_freq='epoch')



epochs = 50
# fine-tune the model
resnet50model_custom.fit_generator(
        train_generator,
        steps_per_epoch=STEP_SIZE_TRAIN,
        epochs=epochs,
        validation_data=validation_generator,
        callbacks=[earlyStop,cp_callback,TB])
```

```
Epoch 1/50
16/16 [==============================] - 531s 32s/step - loss: 0.7392 - acc: 0.4395 -

Epoch 00001: val_acc improved from -inf to 0.37313, saving model to /content/drive/My
Epoch 2/50
16/16 [==============================] - 145s 9s/step - loss: 0.7085 - acc: 0.4435 -

Epoch 00002: val_acc improved from 0.37313 to 0.62687, saving model to /content/drive
Epoch 3/50
16/16 [==============================] - 147s 10s/step - loss: 0.6628 - acc: 0.6250 -

Epoch 00003: val_acc did not improve from 0.62687
Epoch 4/50
16/16 [==============================] - 134s 9s/step - loss: 0.6550 - acc: 0.6290 -

Epoch 00004: val_acc did not improve from 0.62687
Epoch 5/50
16/16 [==============================] - 127s 8s/step - loss: 0.6537 - acc: 0.6210 -

Epoch 00005: val_acc did not improve from 0.62687
Epoch 6/50
16/16 [==============================] - 129s 8s/step - loss: 0.6701 - acc: 0.6008 -

Epoch 00006: val_acc did not improve from 0.62687
Epoch 7/50
16/16 [==============================] - 124s 8s/step - loss: 0.6508 - acc: 0.6411 -
```

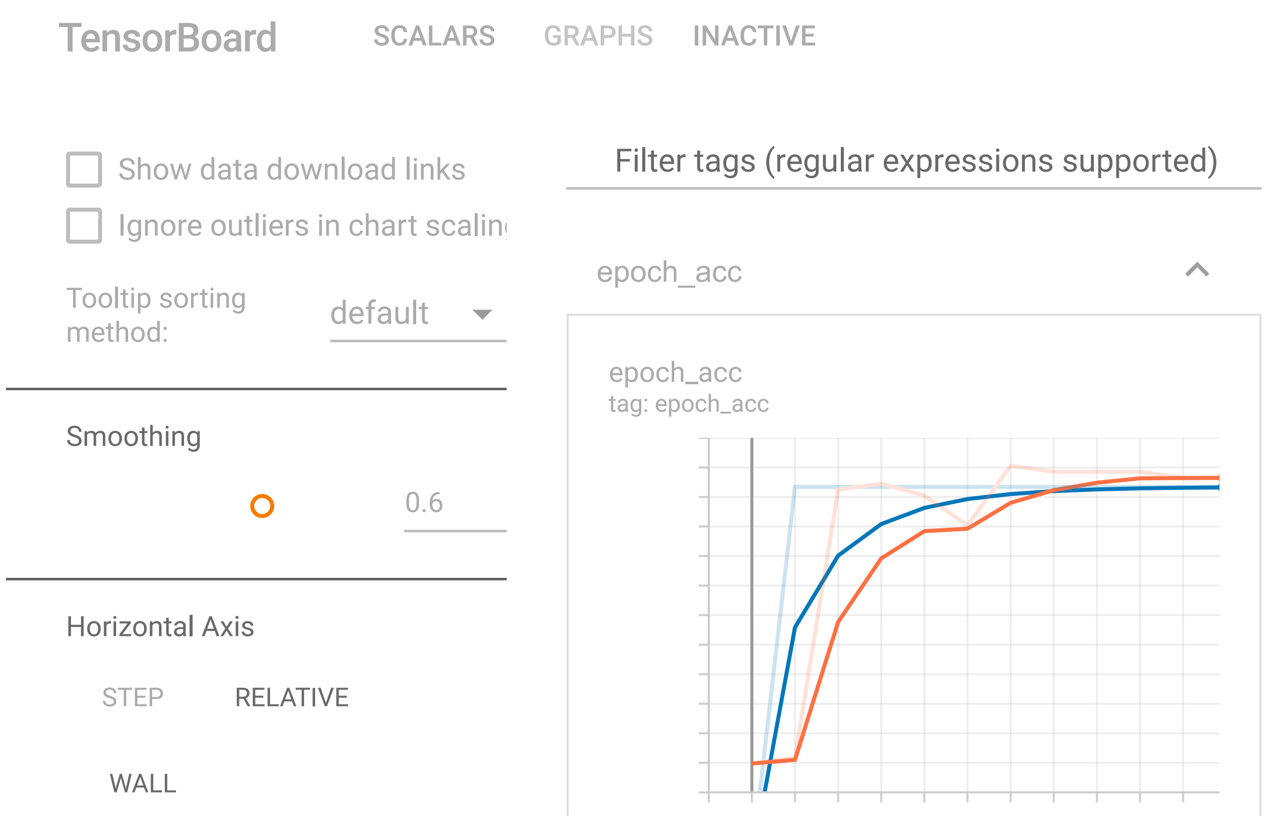
```
Epoch 00007: val_acc did not improve from 0.62687
Epoch 8/50
16/16 [==============================] - 126s 8s/step - loss: 0.6512 - acc: 0.6371 -

Epoch 00008: val_acc did not improve from 0.62687
Epoch 9/50
16/16 [==============================] - 127s 8s/step - loss: 0.6428 - acc: 0.6371 -

Epoch 00009: val_acc did not improve from 0.62687
Epoch 10/50
16/16 [==============================] - 129s 8s/step - loss: 0.6433 - acc: 0.6371 -

Epoch 00010: val_acc did not improve from 0.62687
Epoch 11/50
16/16 [==============================] - 130s 8s/step - loss: 0.6331 - acc: 0.6331 -

Epoch 00011: val_acc did not improve from 0.62687
Epoch 12/50
16/16 [==============================] - 129s 8s/step - loss: 0.6242 - acc: 0.6331 -
Restoring model weights from the end of the best epoch.

Epoch 00012: val_acc did not improve from 0.62687
Epoch 00012: early stopping
<keras.callbacks.History at 0x7f364a40d2d0>
```

```
resnet50model_custom.save(rootdir+'/resnet50model_custom')
```

```
INFO:tensorflow:Assets written to: /content/drive/MyDrive/Colab Notebooks/29. Identi
```

```
%reload_ext tensorboard
```

```
%tensorboard --logdir '/content/drive/MyDrive/Colab Notebooks/29. Identification of Van Go
```



```
#Use the 'dense' layer of the model to generate 1000 features for the patch
custom_resnet_model = Model(inputs = resnet50model_custom.input, outputs=resnet50model_cus

custom_resnet_model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_2[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_co
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_bn
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1_1_re
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_co
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_bn
__________________________________________________________________________________________________
```

```
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0][0]

conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1_2_re

conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_0_co

conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_3_co

conv2_block1_add (Add)          (None, 56, 56, 256)  0           conv2_block1_0_bn
                                                                 conv2_block1_3_bn

conv2_block1_out (Activation)   (None, 56, 56, 256)  0           conv2_block1_add[

conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block1_out[

conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_1_co

conv2_block2_1_relu (Activation (None, 56, 56, 64)   0           conv2_block2_1_bn

conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block2_1_re

conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_2_co

conv2_block2_2_relu (Activation (None, 56, 56, 64)   0           conv2_block2_2_bn

conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block2_2_re

conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block2_3_co
```

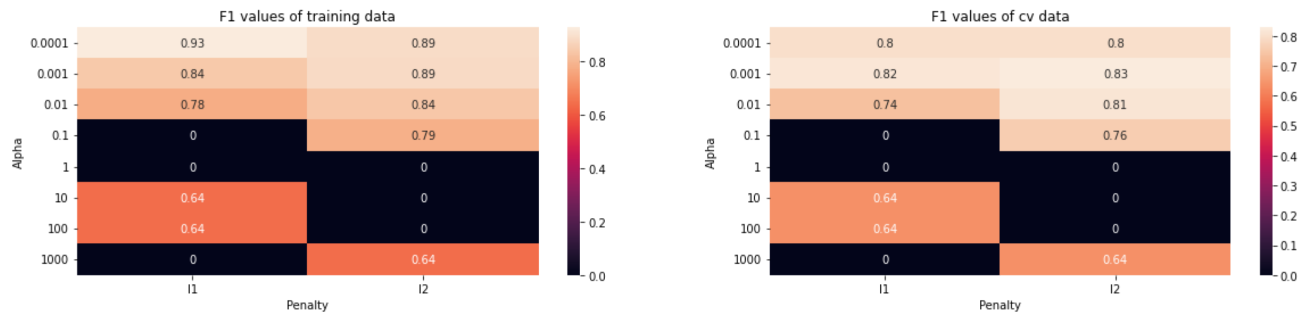
```python
#Pre-process input and predict the output. The output is a list of 1000 features for each
def get_patch_feature_customresnet(patch):

    patch_input = np.expand_dims(patch, axis = 0)            #add an extra dimension for
    patch_preprocessed_input = preprocess_input(patch_input)

    p_feature = custom_resnet_model.predict(patch_preprocessed_input)
    p_feature = p_feature.reshape(1000)

    return p_feature

patch_size = 224                                #patch_size for vgg19 model

X_train = []
Y_train = []

#Get the X, Y array

def get_x_y_arr_customresnet(filename_list,dir_name,y_value,no_of_patches):

  x = []
  y = []

  for file_name in filename_list:

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)
```

```
        for patch in patches:
            patch_feature = get_patch_feature_customresnet(patch)
            x.append(patch_feature)
            y.append(y_value)

    return  x,y


X_train_vg1, Y_train_vg1 = get_x_y_arr_customresnet(train_vg_data,traindir_vg,1,30)
X_train_nvg1, Y_train_nvg1 = get_x_y_arr_customresnet(train_nvg_data,traindir_nvg,0,20)

X_train1 = []
X_train1.extend(X_train_vg1)
X_train1.extend(X_train_nvg1)

Y_train1 = []
Y_train1.extend(Y_train_vg1)
Y_train1.extend(Y_train_nvg1)


X_train1, Y_train1 = shuffle(X_train1, Y_train1)
X_train1 = csr_matrix(X_train1)

X_tr, X_cv, Y_tr, Y_cv = train_test_split(X_train1, Y_train1, test_size = 0.2, random_stat
print("Shape of training data, CV data is :", X_tr.shape, X_cv.shape)
```

```
     Shape of training data, CV data is : (5016, 1000) (1254, 1000)
```

SVM

```
#Hyperparameter tuning

alpha = [10**i for i in range(-4, 4)]
penalty = ['l1', 'l2']

f1_score_tr = []
f1_score_cv = []

for i in alpha:
    for j in penalty:

        clf = SGDClassifier(alpha = i, penalty = j, loss = 'hinge')
        clf.fit(X_tr, Y_tr)

        pred_tr = clf.predict(X_tr)
        pred_cv = clf.predict(X_cv)

        f1_score_tr.append(f1_score(Y_tr, pred_tr))
        f1_score_cv.append(f1_score(Y_cv, pred_cv))


plt.figure(figsize = (20,4))
plt.subplot(1, 2, 1)
f1_score_tr = np.array(f1_score_tr)
f1_score_tr = f1_score_tr.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_tr, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of training data')

plt.subplot(1, 2, 2)
f1_score_cv = np.array(f1_score_cv)
f1_score_cv = f1_score_cv.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_cv, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of cv data')
plt.show()
```

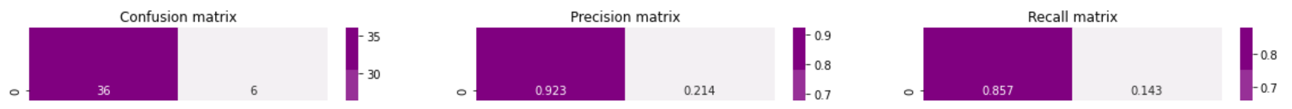


```
#Train with the best parameters

clf = SGDClassifier(alpha = 0.0001, penalty = 'l2', loss = 'hinge')
clf.fit(X_tr, Y_tr)

patch_size = 224
no_of_patches = 20
y = []
pred_vals = []

#Get the X,Y array for test dataset
def get_x_y_arr_probabs(filename_list,dir_name,y_value):

  for file_name in filename_list:
    patch_pred = []

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)

    for patch in patches:
        patch_feature = get_patch_feature_customresnet(patch)
        pred_proba = clf.decision_function([patch_feature])                    #predict pr
        patch_pred.append(pred_proba)


    pred_vals.append(patch_pred)
    y.append(y_value)

  return  pred_vals,y

get_x_y_arr_probabs(test_vg_data,testdir_vg,1)
a,b = get_x_y_arr_probabs(test_nvg_data,testdir_nvg,0)



y_pred_mean = []
y_pred_median = []
y_pred_far = []

for item in pred_vals:
  y_pred_median.append(agg_pred_median(item))
  y_pred_mean.append(agg_pred_mean(item))
  y_pred_far.append(agg_pred_far(item))


print("F1 scores with different Fusion methods")
print("="*200)
print("F1 score for test data - Median is ", f1_score(y, y_pred_median))
print("F1 score for test data - Mean is", f1_score(y, y_pred_mean))
print("F1 score for test data - Far is", f1_score(y, y_pred_far))
```

```
F1 scores with different Fusion methods
==========================================================================================
F1 score for test data - Median is  0.830188679245283
F1 score for test data - Mean is 0.8076923076923077
F1 score for test data - Far is 0.8333333333333333
```

```
f1_test_resnetc_svm = f1_score(y, y_pred_median)

print("F1 score for test data is", f1_test_resnetc_svm)

plot_confusion_matrix(y, y_pred_median)
```

```
F1 score for test data is 0.830188679245283
```



EfficientNet

```
pip install efficientnet
```

```
Requirement already satisfied: efficientnet in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: keras-applications<=1.0.8,>=1.0.7 in /usr/local/lib/py
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cy
```

```
eff_basemodel = EfficientNetB0(include_top=False, weights='imagenet',input_shape=(224,224,
```

```
eff_basemodel.summary()
```

```
Model: "efficientnetb0"
________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
input_3 (InputLayer)            [(None, 224, 224, 3) 0
________________________________________________________________________________
rescaling_2 (Rescaling)         (None, 224, 224, 3)  0           input_3[0][0]
________________________________________________________________________________
normalization_2 (Normalization) (None, 224, 224, 3)  7           rescaling_2[0][0]
________________________________________________________________________________
stem_conv_pad (ZeroPadding2D)   (None, 225, 225, 3)  0           normalization_2[0
________________________________________________________________________________
stem_conv (Conv2D)              (None, 112, 112, 32) 864         stem_conv_pad[0][
________________________________________________________________________________
stem_bn (BatchNormalization)    (None, 112, 112, 32) 128         stem_conv[0][0]
________________________________________________________________________________
stem_activation (Activation)    (None, 112, 112, 32) 0           stem_bn[0][0]
________________________________________________________________________________
block1a_dwconv (DepthwiseConv2D (None, 112, 112, 32) 288         stem_activation[0
________________________________________________________________________________
block1a_bn (BatchNormalization) (None, 112, 112, 32) 128         block1a_dwconv[0]
________________________________________________________________________________
block1a_activation (Activation) (None, 112, 112, 32) 0           block1a_bn[0][0]
________________________________________________________________________________
```

```
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activatio
________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeez
________________________________________________________________________________
block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshap
________________________________________________________________________________
block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce
________________________________________________________________________________
block1a_se_excite (Multiply)    (None, 112, 112, 32) 0           block1a_activatio
                                                                 block1a_se_expand
________________________________________________________________________________
block1a_project_conv (Conv2D)   (None, 112, 112, 16) 512         block1a_se_excite
________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c
________________________________________________________________________________
block2a_expand_conv (Conv2D)    (None, 112, 112, 96) 1536        block1a_project_b
________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co
________________________________________________________________________________
block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn
________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac
________________________________________________________________________________
block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa
________________________________________________________________________________
block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]
________________________________________________________________________________
block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]
________________________________________________________________________________
block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio
________________________________________________________________________________
block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeez
________________________________________________________________________________
block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshap
```

```python
#Create datagenerators with augmentation
batch_size = 16
img_height = 224
img_width = 224


train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    width_shift_range=0.3,
    height_shift_range=0.3,
    rotation_range=30,
    shear_range=0.5,
    zoom_range=.7,
    channel_shift_range=0.3,
    cval=0.5,
    vertical_flip=False,
    brightness_range=[0.1,0.7],
    fill_mode='nearest')

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
        traindir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        testdir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='binary')
```

```
Found 264 images belonging to 2 classes.
Found 67 images belonging to 2 classes.
```

```
x = eff_basemodel.output
x = Dense(1, activation='sigmoid')(x)
efficient_model = Model(inputs=eff_basemodel.input, outputs=x)

for layer in efficient_model.layers[:-526]:
    layer.trainable = False

for layer in efficient_model.layers[-526:]:
    layer.trainable = True
```

```
efficient_model.summary()
```

```
_____________________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activatio
_____________________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeez
_____________________________________________________________________________________________
block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshap

_____________________________________________________________________________________________
block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce
_____________________________________________________________________________________________
block1a_se_excite (Multiply)    (None, 112, 112, 32) 0           block1a_activatio
                                                                 block1a_se_expand
_____________________________________________________________________________________________
block1a_project_conv (Conv2D)   (None, 112, 112, 16) 512         block1a_se_excite
_____________________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c
_____________________________________________________________________________________________
block2a_expand_conv (Conv2D)    (None, 112, 112, 96) 1536        block1a_project_b
_____________________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co
_____________________________________________________________________________________________
block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn
_____________________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac
_____________________________________________________________________________________________
block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa
_____________________________________________________________________________________________
block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]
_____________________________________________________________________________________________
block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]
```

```
block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio

block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeez

block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshap

block2a_se_expand (Conv2D)      (None, 1, 1, 96)     480         block2a_se_reduce

block2a_se_excite (Multiply)    (None, 56, 56, 96)   0           block2a_activatio
                                                                 block2a_se_expand

block2a_project_conv (Conv2D)   (None, 56, 56, 24)   2304        block2a_se_excite

block2a_project_bn (BatchNormal (None, 56, 56, 24)   96          block2a_project_c

block2b_expand_conv (Conv2D)    (None, 56, 56, 144)  3456        block2a_project_b

block2b_expand_bn (BatchNormali (None, 56, 56, 144)  576         block2b_expand_co

block2b_expand_activation (Acti (None, 56, 56, 144)  0           block2b_expand_bn

block2b_dwconv (DepthwiseConv2D (None, 56, 56, 144)  1296        block2b_expand_ac

block2b_bn (BatchNormalization) (None, 56, 56, 144)  576         block2b_dwconv[0]

block2b_activation (Activation) (None, 56, 56, 144)  0           block2b_bn[0][0]

block2b_se_squeeze (GlobalAvera (None, 144)          0           block2b_activatio
```

```
efficient_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.0001), loss='binary_crosse
```

```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=validation_generator.n//validation_generator.batch_size
```

```
earlyStop = EarlyStopping(monitor='val_accuracy', patience=10, restore_best_weights=True,
cp_callback  = ModelCheckpoint(filepath=rootdir+'efficientnet_model.hdf5', monitor='val_ac
TB = TensorBoard(log_dir=rootdir+'/logs/efficientnet_model/'+datetime.datetime.now().strft
                                          embeddings_freq=0, embeddings_layer_names=None,
                                          embeddings_metadata=None, embeddings_data=None,
                                          update_freq='epoch')
rlrop = ReduceLROnPlateau(monitor='val_loss', mode='min', patience= 5, factor= 0.5, min_lr
```

```
epochs = 15
# fine-tune the model
efficient_model.fit_generator(
        train_generator,
        steps_per_epoch=STEP_SIZE_TRAIN,
        epochs=epochs,
        validation_data=validation_generator,
        callbacks=[earlyStop,cp_callback,TB,rlrop])
```

```
Epoch 1/15
```

```
16/16 [==============================] - 149s 10s/step - loss: 0.2085 - accuracy: 0.9

Epoch 00001: val_accuracy improved from -inf to 0.79104, saving model to /content/dri
Epoch 2/15
16/16 [==============================] - 140s 9s/step - loss: 0.2130 - accuracy: 0.93

Epoch 00002: val_accuracy did not improve from 0.79104
Epoch 3/15
16/16 [==============================] - 139s 9s/step - loss: 0.2010 - accuracy: 0.93

Epoch 00003: val_accuracy did not improve from 0.79104
Epoch 4/15
16/16 [==============================] - 141s 9s/step - loss: 0.2281 - accuracy: 0.91

Epoch 00004: val_accuracy did not improve from 0.79104
Epoch 5/15
16/16 [==============================] - 143s 9s/step - loss: 0.1885 - accuracy: 0.91

Epoch 00005: val_accuracy did not improve from 0.79104
Epoch 6/15
16/16 [==============================] - 142s 9s/step - loss: 0.2074 - accuracy: 0.93

Epoch 00006: val_accuracy did not improve from 0.79104

Epoch 00006: ReduceLROnPlateau reducing learning rate to 1.249999968422344e-05.
Epoch 7/15
16/16 [==============================] - ETA: 0s - loss: 0.1836 - accuracy: 0.9315
```

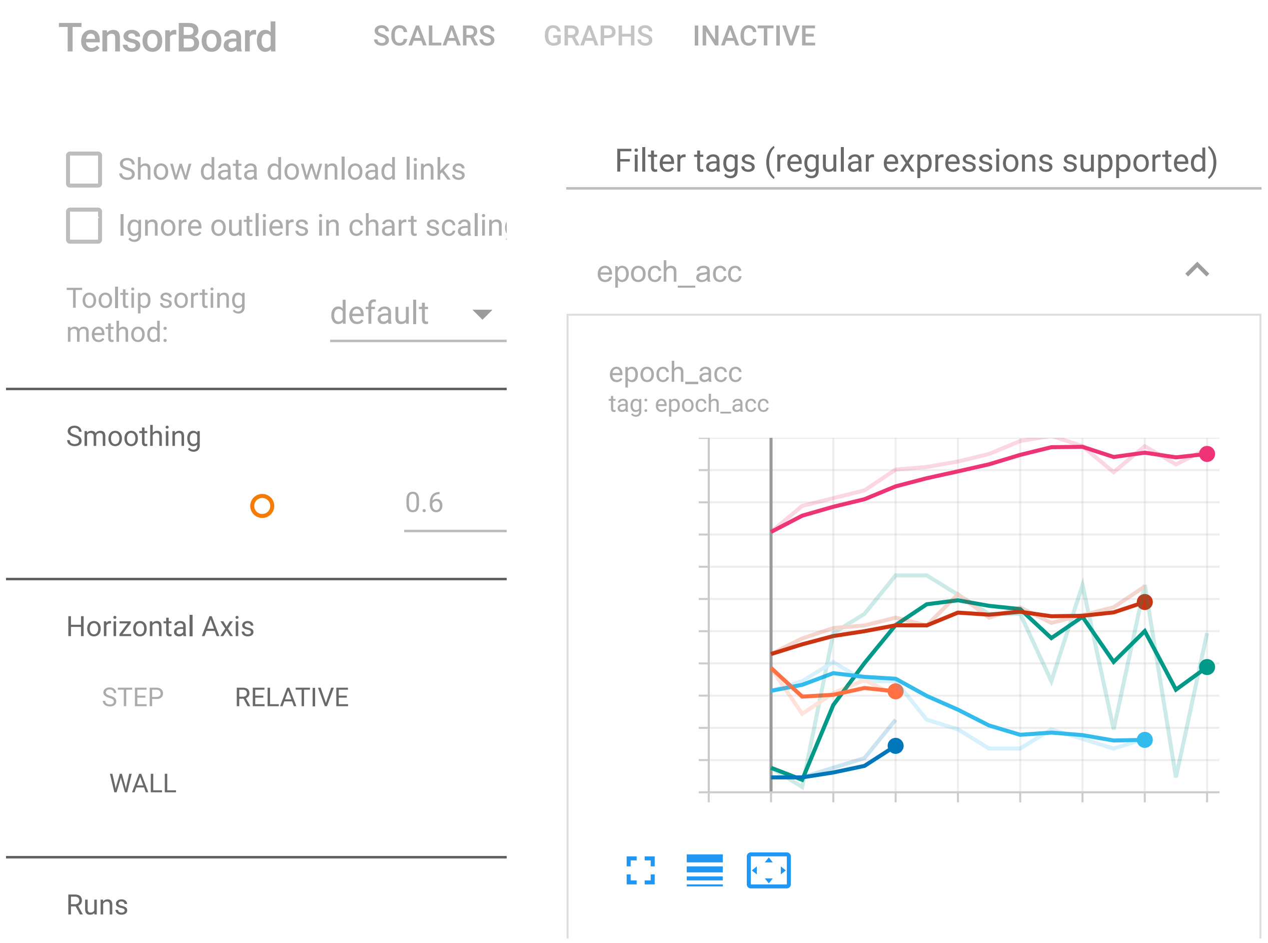
```
efficient_model.save(rootdir+'/efficient_model')
```

```
INFO:tensorflow:Assets written to: /content/drive/MyDrive/Colab Notebooks/29. Identif
```

```
efficient_model = load_model(rootdir+'/efficient_model')
```

```
WARNING:absl:Importing a function (__inference_block3a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block3a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference__wrapped_model_94127) with ops with
WARNING:absl:Importing a function (__inference_block7a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block3a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block2a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_top_activation_layer_call_and_retur
WARNING:absl:Importing a function (__inference_block4c_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block2a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block3b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block4b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block1a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block5b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block4b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block3b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block7a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block2a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block5a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block5a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block3b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block1a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_stem_activation_layer_call_and_retu
WARNING:absl:Importing a function (__inference_block2b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_top_activation_layer_call_and_retur
WARNING:absl:Importing a function (__inference_block6a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block4b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block1a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block4c_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block4c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6c_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block7a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6d_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block3b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5c_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block2a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6d_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6c_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block3a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block5b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block7a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block4a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_model_layer_call_and_return_conditi
WARNING:absl:Importing a function (__inference_block6d_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block2b_activation_layer_call_and_r
```

```
%reload_ext tensorboard
%tensorboard --logdir '/content/drive/MyDrive/Colab Notebooks/29. Identification of Van Go
```



TensorBoard SCALARS GRAPHS INACTIVE

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis STEP RELATIVE WALL

Runs

Filter tags (regular expressions supported)

epoch_acc

epoch_acc
tag: epoch_acc

```
custom_efficientnet_m = Model(inputs = efficient_model.input, outputs=efficient_model.get_

custom_efficientnet_m.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
rescaling_2 (Rescaling)         (None, 224, 224, 3)  0           input_3[0][0]
__________________________________________________________________________________________________
normalization_2 (Normalization) (None, 224, 224, 3)  7           rescaling_2[0][0]
__________________________________________________________________________________________________
stem_conv_pad (ZeroPadding2D)   (None, 225, 225, 3)  0           normalization_2[0
__________________________________________________________________________________________________
stem_conv (Conv2D)              (None, 112, 112, 32) 864         stem_conv_pad[0][
__________________________________________________________________________________________________
stem_bn (BatchNormalization)    (None, 112, 112, 32) 128         stem_conv[0][0]
__________________________________________________________________________________________________
stem_activation (Activation)    (None, 112, 112, 32) 0           stem_bn[0][0]
__________________________________________________________________________________________________
block1a_dwconv (DepthwiseConv2D (None, 112, 112, 32) 288         stem_activation[0
__________________________________________________________________________________________________
block1a_bn (BatchNormalization) (None, 112, 112, 32) 128         block1a_dwconv[0]
__________________________________________________________________________________________________
block1a_activation (Activation) (None, 112, 112, 32) 0           block1a_bn[0][0]
__________________________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activatio
__________________________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeez
```

```
block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshap

block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce

block1a_se_excite (Multiply)    (None, 112, 112, 32) 0           block1a_activatio
                                                                 block1a_se_expand

block1a_project_conv (Conv2D)   (None, 112, 112, 16) 512         block1a_se_excite

block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c

block2a_expand_conv (Conv2D)    (None, 112, 112, 96) 1536        block1a_project_b

block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co

block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn

block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac

block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa

block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]

block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]

block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio

block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeez

block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshap
```

```
def get_patch_feature_efficientNet(patch):

    patch_input = np.expand_dims(patch, axis = 0)             #add an extra dimension for
    patch_preprocessed_input = preprocess_input(patch_input)

    p_feature = custom_efficientnet_m.predict(patch_preprocessed_input)
    p_feature = p_feature.reshape(62720)

    return p_feature

patch_size = 224                             #patch_size for vgg19 model
#no_of_patches = 20                           #No of patches to be generated
X_train = []
Y_train = []

def get_x_y_arr_efficientNet(filename_list,dir_name,y_value,no_of_patches):

  x = []
  y = []

  for file_name in filename_list:

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)

    for patch in patches:
        patch_feature = get_patch_feature_efficientNet(patch)
        x.append(patch_feature)
        y.append(y_value)

  return  x,y


'''This function is used to generate patches for a particular image.
The number of patches and patch_size are passed as function parameters.
The function generates the number of patches as the given parameter'''
def get_patches_for_img(img_path, patch_size, no_of_patches):

    patches = []

    #read the image at the given path
    img = cv2.imread(img_path, 1)
    image_height = img.shape[0]
    image_width = img.shape[1]

    #Subtract patch_size from image's height and width to avoid out of bounds error
    range_x = image_height - patch_size
    range_y = image_width - patch_size

    #Generate patches for each image. The number of patches are passed as parameter.
    for i in range(no_of_patches):

        #Generate patch from random area of the image
        x = np.random.randint(low = 0, high = range_x)
        y = np.random.randint(low = 0, high = range_y)

        #The patch is calculated by adding the patch_size to both x and y co-ordinates
        patch = img[x : x+patch_size, y : y+patch_size, :]
        patches.append(patch)

    return patches



X_train_vg1, Y_train_vg1 = get_x_y_arr_efficientNet(train_vg_data,traindir_vg,1,30)
X_train_nvg1, Y_train_nvg1 = get_x_y_arr_efficientNet(train_nvg_data,traindir_nvg,0,20)

X_train1 = []
X_train1.extend(X_train_vg1)
X_train1.extend(X_train_nvg1)

Y_train1 = []
Y_train1.extend(Y_train_vg1)
Y_train1.extend(Y_train_nvg1)


X_train1, Y_train1 = shuffle(X_train1, Y_train1)
X_train1 = csr_matrix(X_train1)

X_tr, X_cv, Y_tr, Y_cv = train_test_split(X_train1, Y_train1, test_size = 0.2, random_stat
print("Shape of training data, CV data is :", X_tr.shape, X_cv.shape)
```

```
Shape of training data, CV data is : (5016, 62720) (1254, 62720)
```

```
alpha = [10**i for i in range(-4, 1)]
penalty = ['l2']

f1_score_tr = []
f1_score_cv = []

for i in alpha:
    print('Processing for alpha = ', i)
    for j in penalty:

        clf = SGDClassifier(alpha = i, penalty = j, loss = 'hinge')
        clf.fit(X_tr, Y_tr)

        pred_tr = clf.predict(X_tr)
        pred_cv = clf.predict(X_cv)

        f1_score_tr.append(f1_score(Y_tr, pred_tr))
        f1_score_cv.append(f1_score(Y_cv, pred_cv))

plt.figure(figsize = (20,4))
plt.subplot(1, 2, 1)
f1_score_tr = np.array(f1_score_tr)
f1_score_tr = f1_score_tr.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_tr, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of training data')

plt.subplot(1, 2, 2)
f1_score_cv = np.array(f1_score_cv)
f1_score_cv = f1_score_cv.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_cv, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of cv data')
plt.show()
```

```
Processing for alpha =  [0.0001, 0.001, 0.01, 0.1, 1]
```

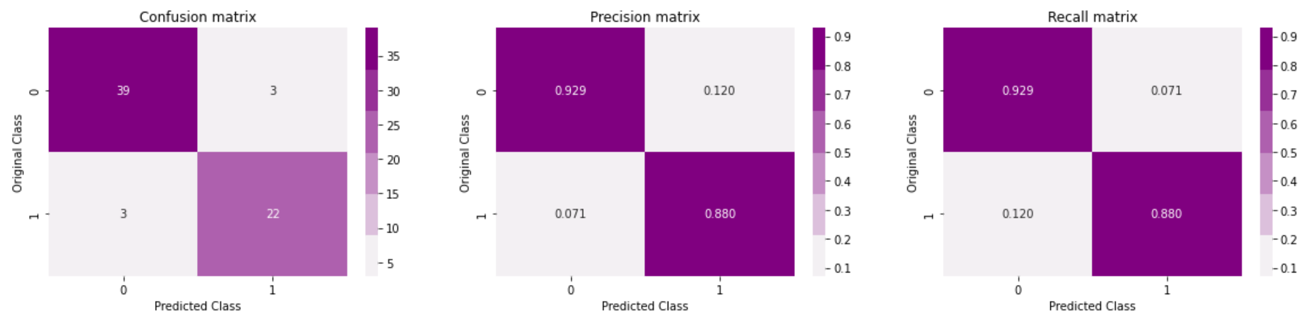
```
clf = SGDClassifier(alpha = 1, penalty = 'l2', loss = 'hinge')
clf.fit(X_tr, Y_tr)

patch_size = 224
no_of_patches = 20
y = []
pred_vals = []

def get_x_y_arr_probabs(filename_list,dir_name,y_value):

  for file_name in filename_list:
    patch_pred = []
```

```
      img_path = dir_name + '/' + file_name
      patches = get_patches_for_img(img_path, patch_size, no_of_patches)

      for patch in patches:
          patch_feature = get_patch_feature_efficientNet(patch)

          pred_proba = clf.decision_function([patch_feature])                    #predict pr
          patch_pred.append(pred_proba)


      pred_vals.append(patch_pred)
      y.append(y_value)

    return  pred_vals,y

get_x_y_arr_probabs(test_vg_data,testdir_vg,1)
a,b = get_x_y_arr_probabs(test_nvg_data,testdir_nvg,0)


y_pred_mean = []
y_pred_median = []
y_pred_far = []

for item in pred_vals:
  y_pred_median.append(agg_pred_median(item))
  y_pred_mean.append(agg_pred_mean(item))
  y_pred_far.append(agg_pred_far(item))



print("F1 scores with different Fusion methods")
print("="*200)
print("F1 score for test data - Median is ", f1_score(y, y_pred_median))
print("F1 score for test data - Mean is", f1_score(y, y_pred_mean))
print("F1 score for test data - Far is", f1_score(y, y_pred_far))
```

```
F1 scores with different Fusion methods
=====================================================================================
F1 score for test data - Median is  0.8461538461538461
F1 score for test data - Mean is 0.88
F1 score for test data - Far is 0.88
```

```
f1_test_eff_svm = f1_score(y, y_pred_mean)

print("F1 score for test data is", f1_test_eff_svm)
plot_confusion_matrix(y, y_pred_mean)
```

```
F1 score for test data is 0.88
```



EfficientNet with custom layers

```
eff_basemodel = EfficientNetB0(include_top=False, weights='imagenet',input_shape=(224,224,
eff_basemodel.summary()
```

```
block1a_bn (BatchNormalization) (None, 112, 112, 32) 128         block1a_dwconv[0]
_______________________________________________________________________________________
block1a_activation (Activation) (None, 112, 112, 32) 0           block1a_bn[0][0]
_______________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activatio
_______________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeez
_______________________________________________________________________________________
block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshap
_______________________________________________________________________________________
block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce
_______________________________________________________________________________________
block1a_se_excite (Multiply)    (None, 112, 112, 32) 0           block1a_activatio
                                                                 block1a_se_expand
_______________________________________________________________________________________
block1a_project_conv (Conv2D)   (None, 112, 112, 16) 512         block1a_se_excite
_______________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c

_______________________________________________________________________________________
block2a_expand_conv (Conv2D)    (None, 112, 112, 96) 1536        block1a_project_b
_______________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co
_______________________________________________________________________________________
block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn
_______________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac
_______________________________________________________________________________________
block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa
_______________________________________________________________________________________
block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]
_______________________________________________________________________________________
block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]
_______________________________________________________________________________________
block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio
_______________________________________________________________________________________
block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeez
_______________________________________________________________________________________
block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshap
_______________________________________________________________________________________
block2a_se_expand (Conv2D)      (None, 1, 1, 96)     480         block2a_se_reduce
_______________________________________________________________________________________
block2a_se_excite (Multiply)    (None, 56, 56, 96)   0           block2a_activatio
```

```
block2a_se_excite (Multiply)    (None, 56, 56, 96)   0           block2a_activatio
                                                                  block2a_se_expand

block2a_project_conv (Conv2D)   (None, 56, 56, 24)   2304        block2a_se_excite

block2a_project_bn (BatchNormal (None, 56, 56, 24)   96          block2a_project_c

block2b_expand_conv (Conv2D)    (None, 56, 56, 144)  3456        block2a_project_b

block2b_expand_bn (BatchNormali (None, 56, 56, 144)  576         block2b_expand_co

block2b_expand_activation (Acti (None, 56, 56, 144)  0           block2b_expand_bn

block2b_dwconv (DepthwiseConv2D (None, 56, 56, 144)  1296        block2b_expand_ac

block2b_bn (BatchNormalization) (None, 56, 56, 144)  576         block2b_dwconv[0]
```

```
#Create datagenerators with augmentation

from efficientnet.tfkeras import preprocess_input
from efficientnet import model
from keras.models import Model, load_model

batch_size = 16
img_height = 224
img_width = 224


train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    width_shift_range=0.3,
    height_shift_range=0.3,
    rotation_range=30,
    shear_range=0.5,
    zoom_range=.7,
    channel_shift_range=0.3,
    cval=0.5,
    vertical_flip=False,
    brightness_range=[0.1,0.7],
    fill_mode='nearest')

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
        traindir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        testdir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='binary')
```

```
Found 264 images belonging to 2 classes.
Found 67 images belonging to 2 classes.
```

```
x = eff_basemodel.get_layer('top_conv').output

custom_conv=Conv2D(filters=128, kernel_size=1, padding="same", activation="relu")(x)
b1 =  BatchNormalization()(custom_conv)
a1 = Activation('relu')(b1)

top_fc1 = Flatten()(a1)
d2 = Dense(1000,activation='relu')(top_fc1)
op = Dense(1, activation='sigmoid')(d2)

efficient_model = Model(inputs=eff_basemodel.input, outputs=op)

for layer in efficient_model.layers[:-526]:
    layer.trainable = False

for layer in efficient_model.layers[-526:]:
    layer.trainable = True
```

```
efficient_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.0001), loss='binary_crosse
```

```
efficient_model.summary()
```

```
Model: "model"
________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
________________________________________________________________________________
rescaling (Rescaling)           (None, 224, 224, 3)  0           input_1[0][0]
________________________________________________________________________________
normalization (Normalization)   (None, 224, 224, 3)  7           rescaling[0][0]
________________________________________________________________________________
stem_conv_pad (ZeroPadding2D)   (None, 225, 225, 3)  0           normalization[0][
________________________________________________________________________________
stem_conv (Conv2D)              (None, 112, 112, 32) 864         stem_conv_pad[0][
________________________________________________________________________________
stem_bn (BatchNormalization)    (None, 112, 112, 32) 128         stem_conv[0][0]
________________________________________________________________________________
stem_activation (Activation)    (None, 112, 112, 32) 0           stem_bn[0][0]
________________________________________________________________________________
block1a_dwconv (DepthwiseConv2D (None, 112, 112, 32) 288         stem_activation[0
________________________________________________________________________________
block1a_bn (BatchNormalization) (None, 112, 112, 32) 128         block1a_dwconv[0]
________________________________________________________________________________
block1a_activation (Activation) (None, 112, 112, 32) 0           block1a_bn[0][0]
________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activatio
________________________________________________________________________________
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeez
________________________________________________________________________________
block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshap
________________________________________________________________________________
```

```
block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce
____________________________________________________________________________________
block1a_se_excite (Multiply)    (None, 112, 112, 32) 0           block1a_activatio
                                                                 block1a_se_expand
____________________________________________________________________________________
block1a_project_conv (Conv2D)   (None, 112, 112, 16) 512         block1a_se_excite
____________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c
____________________________________________________________________________________
block2a_expand_conv (Conv2D)    (None, 112, 112, 96) 1536        block1a_project_b
____________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co
____________________________________________________________________________________
block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn
____________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac
____________________________________________________________________________________
block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa
____________________________________________________________________________________
block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]
____________________________________________________________________________________
block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]
____________________________________________________________________________________
block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio
____________________________________________________________________________________
block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeez
____________________________________________________________________________________
block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshap
```

```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=validation_generator.n//validation_generator.batch_size

earlyStop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True, verb
cp_callback  = ModelCheckpoint(filepath=rootdir+'efficientnet_model_custom.hdf5', monitor=
TB = TensorBoard(log_dir=rootdir+'/logs/efficientnet_model_custom/'+datetime.datetime.now(
                                          embeddings_freq=0, embeddings_layer_names=None,
                                          embeddings_metadata=None, embeddings_data=None,
                                          update_freq='epoch')
rlrop = ReduceLROnPlateau(monitor='val_loss', mode='min', patience= 5, factor= 0.5, min_lr


epochs = 15
# fine-tune the model
efficient_model.fit_generator(
        train_generator,
        steps_per_epoch=STEP_SIZE_TRAIN,
        epochs=epochs,
        validation_data=validation_generator,
        callbacks=[earlyStop,cp_callback,TB,rlrop])
```

```
Epoch 1/15
16/16 [==============================] - 562s 34s/step - loss: 0.8633 - accuracy: 

Epoch 00001: val_loss improved from inf to 2.70109, saving model to /content/drive
Epoch 2/15
16/16 [==============================] - 147s 9s/step - loss: 0.5596 - accuracy: 0
```

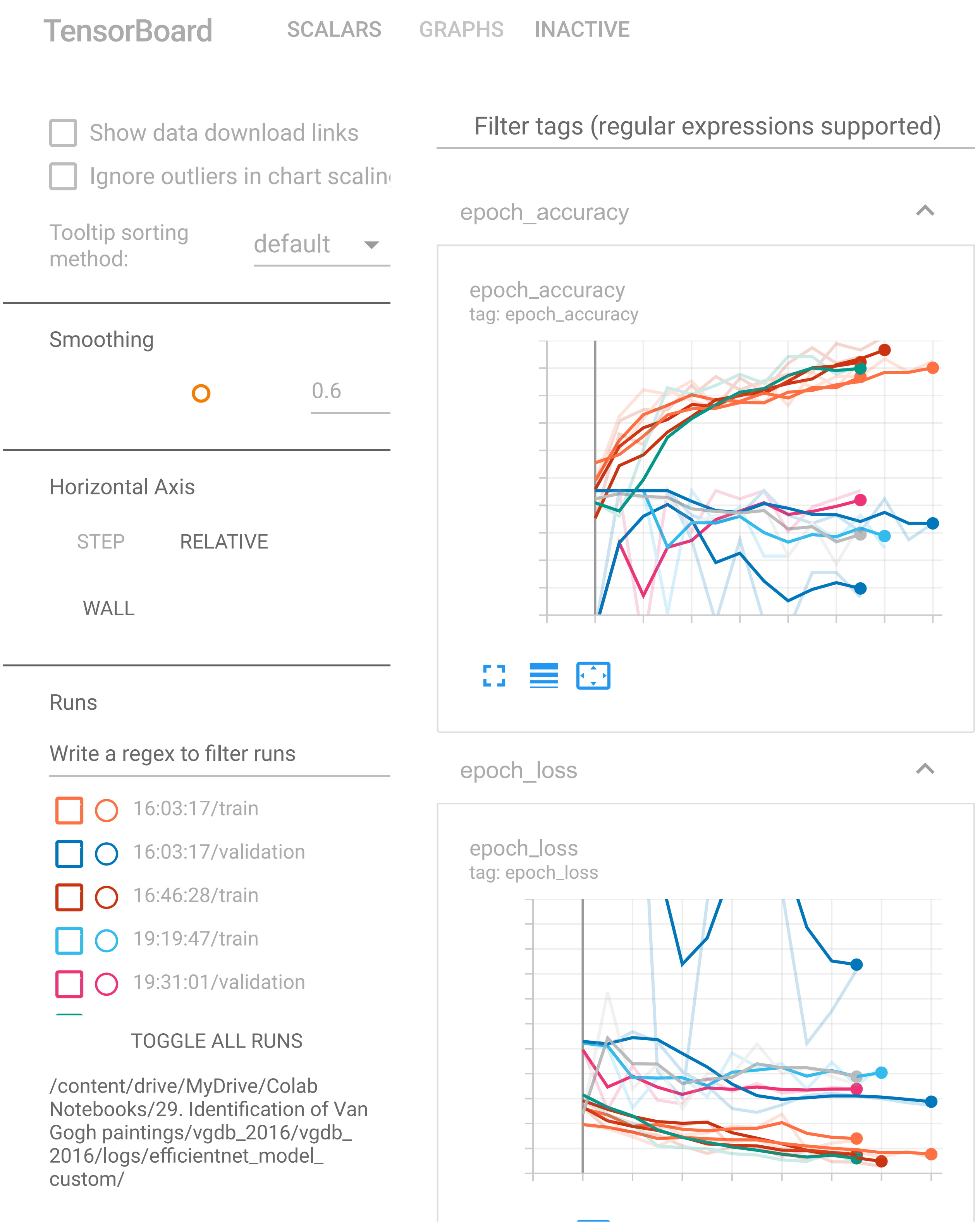
```
Epoch 00002: val_loss improved from 2.70109 to 2.11142, saving model to /content/d
Epoch 3/15
16/16 [==============================] - 142s 9s/step - loss: 0.4960 - accuracy: 0

Epoch 00003: val_loss improved from 2.11142 to 1.77862, saving model to /content/d
Epoch 4/15
16/16 [==============================] - 136s 9s/step - loss: 0.4605 - accuracy: 0

Epoch 00004: val_loss did not improve from 1.77862
Epoch 5/15
16/16 [==============================] - 131s 8s/step - loss: 0.4349 - accuracy: 0

Epoch 00005: val_loss did not improve from 1.77862
Epoch 6/15
16/16 [==============================] - 129s 8s/step - loss: 0.3851 - accuracy: 0

Epoch 00006: val_loss did not improve from 1.77862
Epoch 7/15
16/16 [==============================] - 123s 8s/step - loss: 0.3356 - accuracy: 0

Epoch 00007: val_loss improved from 1.77862 to 1.77663, saving model to /content/d
Epoch 8/15
16/16 [==============================] - 128s 8s/step - loss: 0.3152 - accuracy: 0

Epoch 00008: val_loss improved from 1.77663 to 1.24496, saving model to /content/d
Epoch 9/15
16/16 [==============================] - 128s 8s/step - loss: 0.3529 - accuracy: 0

Epoch 00009: val_loss did not improve from 1.24496
Epoch 10/15
16/16 [==============================] - 128s 8s/step - loss: 0.4414 - accuracy: 0

Epoch 00010: val_loss improved from 1.24496 to 0.91142, saving model to /content/d
Epoch 11/15
16/16 [==============================] - 131s 8s/step - loss: 0.3553 - accuracy: 0

Epoch 00011: val_loss did not improve from 0.91142
Epoch 12/15
16/16 [==============================] - 132s 8s/step - loss: 0.2751 - accuracy: 0

Epoch 00012: val_loss did not improve from 0.91142
Epoch 13/15
16/16 [==============================] - 127s 8s/step - loss: 0.3133 - accuracy: 0

Epoch 00013: val_loss did not improve from 0.91142
Epoch 14/15
16/16 [==============================] - 128s 8s/step - loss: 0.2903 - accuracy: 0

Epoch 00014: val_loss did not improve from 0.91142
Epoch 15/15
16/16 [==============================] - 131s 8s/step - loss: 0.2825 - accuracy: 0
```

```
efficient_model.save(rootdir+'/efficient_model_custom')
```

```
INFO:tensorflow:Assets written to: /content/drive/MyDrive/Colab Notebooks/29. Identi
```

```
%reload_ext tensorboard
```

```
%tensorboard --logdir '/content/drive/MyDrive/Colab Notebooks/29. Identification of Van Go
```



TensorBoard SCALARS GRAPHS INACTIVE

Show data download links
Ignore outliers in chart scalin
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis
STEP RELATIVE
WALL

Runs
Write a regex to filter runs
16:03:17/train
16:03:17/validation
16:46:28/train
19:19:47/train
19:31:01/validation

TOGGLE ALL RUNS

/content/drive/MyDrive/Colab Notebooks/29. Identification of Van Gogh paintings/vgdb_2016/vgdb_2016/logs/efficientnet_model_custom/

Filter tags (regular expressions supported)

epoch_accuracy
epoch_accuracy
tag: epoch_accuracy

epoch_loss
epoch_loss
tag: epoch_loss

```
efficient_model.summary()
```

```
block1a_dwconv (DepthwiseConv2D (None, 112, 112, 32) 288         stem_activation[0
block1a_bn (BatchNormalization) (None, 112, 112, 32) 128         block1a_dwconv[0]
block1a_activation (Activation) (None, 112, 112, 32) 0           block1a_bn[0][0]
block1a_se_squeeze (GlobalAvera (None, 32)           0           block1a_activatio
block1a_se_reshape (Reshape)    (None, 1, 1, 32)     0           block1a_se_squeez

block1a_se_reduce (Conv2D)      (None, 1, 1, 8)      264         block1a_se_reshap

block1a_se_expand (Conv2D)      (None, 1, 1, 32)     288         block1a_se_reduce

block1a_se_excite (Multiply)    (None, 112, 112, 32) 0           block1a_activatio
                                                                 block1a_se_expand

block1a_project_conv (Conv2D)   (None, 112, 112, 16) 512         block1a_se_excite

block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c

block2a_expand_conv (Conv2D)    (None, 112, 112, 96) 1536        block1a_project_b

block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co

block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn

block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac

block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa

block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]

block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]

block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio

block2a_se_reshape (Reshape)    (None, 1, 1, 96)     0           block2a_se_squeez

block2a_se_reduce (Conv2D)      (None, 1, 1, 4)      388         block2a_se_reshap

block2a_se_expand (Conv2D)      (None, 1, 1, 96)     480         block2a_se_reduce

block2a_se_excite (Multiply)    (None, 56, 56, 96)   0           block2a_activatio
                                                                 block2a_se_expand

block2a_project_conv (Conv2D)   (None, 56, 56, 24)   2304        block2a_se_excite

block2a_project_bn (BatchNormal (None, 56, 56, 24)   96          block2a_project_c

block2b_expand_conv (Conv2D)    (None, 56, 56, 144)  3456        block2a_project_b


block2b_expand_bn (BatchNormali (None, 56, 56, 144)  576         block2b_expand_co

block2b_expand_activation (Acti (None, 56, 56, 144)  0           block2b_expand_bn

block2b_dwconv (DepthwiseConv2D (None, 56, 56, 144)  1296        block2b_expand_ac
```

```
efficient_model = load_model(rootdir+'/efficient_model_custom')
```

```
WARNING:absl:Importing a function (__inference_block4c_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6d_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block2b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block2b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block3a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6d_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6d_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block2b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block5c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_model_layer_call_and_return_conditi
WARNING:absl:Importing a function (__inference_block4a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6c_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block2b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block6c_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block1a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block3b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6d_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block1a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block4b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block4a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block2b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block3a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block7a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5c_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6d_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block3b_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block6c_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block1a_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block5a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference__wrapped_model_34900) with ops with
WARNING:absl:Importing a function (__inference_stem_activation_layer_call_and_retu
WARNING:absl:Importing a function (__inference_block2a_expand_activation_layer_cal

WARNING:absl:Importing a function (__inference_block2a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block3a_expand_activation_layer_cal
WARNING:absl:Importing a function (__inference_block5b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block5b_se_reduce_layer_call_and_re
WARNING:absl:Importing a function (__inference_block4b_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block2a_activation_layer_call_and_r
WARNING:absl:Importing a function (__inference_block6a_expand_activation_layer_cal
```

```
custom_efficientnet_m = Model(inputs = efficient_model.input, outputs=efficient_model.get_
```

```
custom_efficientnet_m.summary()
```

```
________________________________________________________________________________
block1a_se_squeeze (GlobalAvera (None, 32)          0           block1a_activatio
________________________________________________________________________________
```

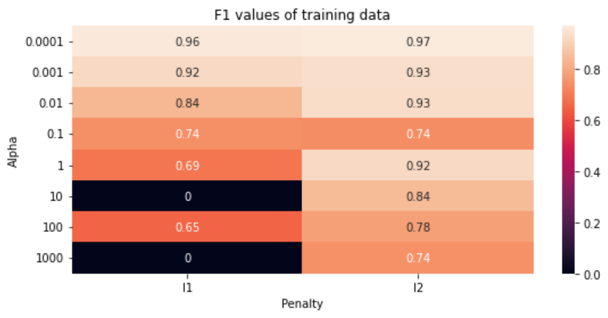
```
block1a_se_reshape (Reshape)     (None, 1, 1, 32)     0           block1a_se_squeez
_______________________________________________________________________________________
block1a_se_reduce (Conv2D)       (None, 1, 1, 8)      264         block1a_se_reshap
_______________________________________________________________________________________
block1a_se_expand (Conv2D)       (None, 1, 1, 32)     288         block1a_se_reduce
_______________________________________________________________________________________
block1a_se_excite (Multiply)     (None, 112, 112, 32) 0           block1a_activatio
                                                                  block1a_se_expand
_______________________________________________________________________________________
block1a_project_conv (Conv2D)    (None, 112, 112, 16) 512         block1a_se_excite
_______________________________________________________________________________________
block1a_project_bn (BatchNormal (None, 112, 112, 16) 64          block1a_project_c
_______________________________________________________________________________________
block2a_expand_conv (Conv2D)     (None, 112, 112, 96) 1536        block1a_project_b
_______________________________________________________________________________________
block2a_expand_bn (BatchNormali (None, 112, 112, 96) 384         block2a_expand_co
_______________________________________________________________________________________
block2a_expand_activation (Acti (None, 112, 112, 96) 0           block2a_expand_bn
_______________________________________________________________________________________
block2a_dwconv_pad (ZeroPadding (None, 113, 113, 96) 0           block2a_expand_ac
_______________________________________________________________________________________
block2a_dwconv (DepthwiseConv2D (None, 56, 56, 96)   864         block2a_dwconv_pa
_______________________________________________________________________________________
block2a_bn (BatchNormalization) (None, 56, 56, 96)   384         block2a_dwconv[0]
_______________________________________________________________________________________
block2a_activation (Activation) (None, 56, 56, 96)   0           block2a_bn[0][0]
_______________________________________________________________________________________
block2a_se_squeeze (GlobalAvera (None, 96)           0           block2a_activatio
_______________________________________________________________________________________
block2a_se_reshape (Reshape)     (None, 1, 1, 96)     0           block2a_se_squeez
_______________________________________________________________________________________
block2a_se_reduce (Conv2D)       (None, 1, 1, 4)      388         block2a_se_reshap
_______________________________________________________________________________________
block2a_se_expand (Conv2D)       (None, 1, 1, 96)     480         block2a_se_reduce
_______________________________________________________________________________________
block2a_se_excite (Multiply)     (None, 56, 56, 96)   0           block2a_activatio
                                                                  block2a_se_expand
_______________________________________________________________________________________
block2a_project_conv (Conv2D)    (None, 56, 56, 24)   2304        block2a_se_excite
_______________________________________________________________________________________
block2a_project_bn (BatchNormal (None, 56, 56, 24)   96          block2a_project_c
_______________________________________________________________________________________
block2b_expand_conv (Conv2D)     (None, 56, 56, 144)  3456        block2a_project_b
_______________________________________________________________________________________
block2b_expand_bn (BatchNormali (None, 56, 56, 144)  576         block2b_expand_co
_______________________________________________________________________________________
block2b_expand_activation (Acti (None, 56, 56, 144)  0           block2b_expand_bn
_______________________________________________________________________________________
block2b_dwconv (DepthwiseConv2D (None, 56, 56, 144)  1296        block2b_expand_ac
_______________________________________________________________________________________
block2b_bn (BatchNormalization) (None, 56, 56, 144)  576         block2b_dwconv[0]
_______________________________________________________________________________________
block2b_activation (Activation) (None, 56, 56, 144)  0           block2b_bn[0][0]
_______________________________________________________________________________________
block2b_se_squeeze (GlobalAvera (None, 144)          0           block2b_activatio
```

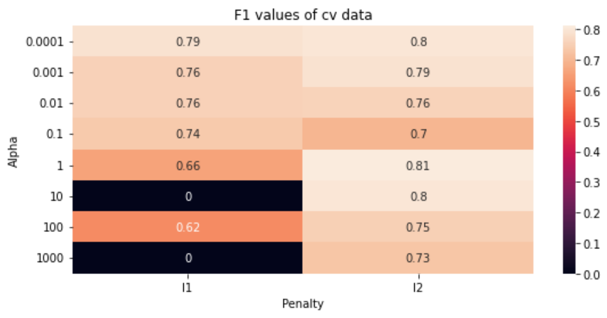
```
def get_patch_feature_efficientNet(patch):

    patch_input = np.expand_dims(patch, axis = 0)           #add an extra dimension for
    patch_preprocessed_input = preprocess_input(patch_input)

    p_feature = custom_efficientnet_m.predict(patch_preprocessed_input)
    #p_feature = p_feature.reshape(62720)
    p_feature = p_feature.reshape(6272)

    return p_feature

patch_size = 224                            #patch_size for vgg19 model
#no_of_patches = 20                          #No of patches to be generated
X_train = []
Y_train = []

def get_x_y_arr_efficientNet(filename_list,dir_name,y_value,no_of_patches):

  x = []
  y = []

  for file_name in filename_list:

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)

    for patch in patches:
        patch_feature = get_patch_feature_efficientNet(patch)
        x.append(patch_feature)
        y.append(y_value)

  return  x,y

'''This function is used to generate patches for a particular image.
The number of patches and patch_size are passed as function parameters.
The function generates the number of patches as the given parameter'''
def get_patches_for_img(img_path, patch_size, no_of_patches):

    patches = []

    #read the image at the given path
    img = cv2.imread(img_path, 1)
    image_height = img.shape[0]
    image_width = img.shape[1]

    #Subtract patch_size from image's height and width to avoid out of bounds error
    range_x = image_height - patch_size
    range_y = image_width - patch_size

    #Generate patches for each image. The number of patches are passed as parameter.
    for i in range(no_of_patches):

        #Generate patch from random area of the image
        x = np.random.randint(low = 0, high = range_x)
        y = np.random.randint(low = 0, high = range_y)

        #The patch is calculated by adding the patch_size to both x and y co-ordinates
        patch = img[x : x+patch_size, y : y+patch_size, :]
        patches.append(patch)

    return patches


X_train_vg1, Y_train_vg1 = get_x_y_arr_efficientNet(train_vg_data,traindir_vg,1,30)
X_train_nvg1, Y_train_nvg1 = get_x_y_arr_efficientNet(train_nvg_data,traindir_nvg,0,20)

X_train1 = []
X_train1.extend(X_train_vg1)
X_train1.extend(X_train_nvg1)

Y_train1 = []
Y_train1.extend(Y_train_vg1)
Y_train1.extend(Y_train_nvg1)

X_train1, Y_train1 = shuffle(X_train1, Y_train1)
X_train1 = csr_matrix(X_train1)

X_tr, X_cv, Y_tr, Y_cv = train_test_split(X_train1, Y_train1, test_size = 0.2, random_stat
print("Shape of training data, CV data is :", X_tr.shape, X_cv.shape)
```

```
     Shape of training data, CV data is : (5016, 6272) (1254, 6272)
```

```
#Hyperparameter tuning
alpha = [10**i for i in range(-4, 4)]
penalty = ['l1', 'l2']

f1_score_tr = []
f1_score_cv = []

for i in alpha:
    for j in penalty:

        clf = SGDClassifier(alpha = i, penalty = j, loss = 'hinge')
        clf.fit(X_tr, Y_tr)

        pred_tr = clf.predict(X_tr)
        pred_cv = clf.predict(X_cv)

        f1_score_tr.append(f1_score(Y_tr, pred_tr))
        f1_score_cv.append(f1_score(Y_cv, pred_cv))



plt.figure(figsize = (20,4))
plt.subplot(1, 2, 1)
f1_score_tr = np.array(f1_score_tr)
f1_score_tr = f1_score_tr.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_tr, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of training data')

plt.subplot(1, 2, 2)
```

```
f1_score_cv = np.array(f1_score_cv)
f1_score_cv = f1_score_cv.reshape(len(alpha),len(penalty))
sns.heatmap(f1_score_cv, annot = True, xticklabels = penalty, yticklabels = alpha)
plt.xlabel('Penalty')
plt.ylabel('Alpha')
plt.title('F1 values of cv data')
plt.show()
```





```
clf = SGDClassifier(alpha = 0.0001, penalty = 'l1', loss = 'hinge')
clf.fit(X_tr, Y_tr)

patch_size = 224
no_of_patches = 20
y = []
pred_vals = []

def get_x_y_arr_probabs(filename_list,dir_name,y_value):

  for file_name in filename_list:
    patch_pred = []

    img_path = dir_name + '/' + file_name
    patches = get_patches_for_img(img_path, patch_size, no_of_patches)

    for patch in patches:
        patch_feature = get_patch_feature_efficientNet(patch)

        pred_proba = clf.decision_function([patch_feature])                  #predict pr
        patch_pred.append(pred_proba)


    pred_vals.append(patch_pred)
    y.append(y_value)

  return  pred_vals,y

get_x_y_arr_probabs(test_vg_data,testdir_vg,1)
a,b = get_x_y_arr_probabs(test_nvg_data,testdir_nvg,0)
```

```
y_pred_mean = []
y_pred_median = []
y_pred_far = []

for item in pred_vals:
  y_pred_median.append(agg_pred_median(item))
  y_pred_mean.append(agg_pred_mean(item))
  y_pred_far.append(agg_pred_far(item))


print("F1 scores with different Fusion methods")
print("="*200)
print("F1 score for test data - Median is ", f1_score(y, y_pred_median))
print("F1 score for test data - Mean is", f1_score(y, y_pred_mean))
print("F1 score for test data - Far is", f1_score(y, y_pred_far))
```

```
F1 scores with different Fusion methods
==========================================================================================
F1 score for test data - Median is  0.7924528301886793
F1 score for test data - Mean is 0.8
F1 score for test data - Far is 0.8275862068965517
```

Final Model Performance Comparison

```
from prettytable import PrettyTable
all_model_results = PrettyTable(["Model","F1-Score : Far", "F1-Score : Mean","F1-Score : M

# Add rows
all_model_results.add_row(["Baseline model",0.55,0.55,0.53])
all_model_results.add_row(["VGG19",0.85,0.88,0.91])
all_model_results.add_row(["VGG16 with custom layers",0.84,0.89,0.87])
all_model_results.add_row(["Resnet",0.87,0.85,0.85])
all_model_results.add_row(["Resnet with custom layers",0.83,0.81,0.83])
all_model_results.add_row(["EfficientNet",0.88,0.88,0.84])
all_model_results.add_row(["EfficientNet with custom layers",0.82,0.8,0.79])

print("The performance of the models")
print(all_model_results)
```

```
The performance of the models
+---------------------------------+----------------+-----------------+---------------
|              Model              | F1-Score : Far | F1-Score : Mean | F1-Score : Me
+---------------------------------+----------------+-----------------+---------------
|          Baseline model         |      0.55      |       0.55      |       0.53
|              VGG19              |      0.85      |       0.88      |       0.91
|     VGG16 with custom layers    |      0.84      |       0.89      |       0.87
|              Resnet             |      0.87      |       0.85      |       0.85
|    Resnet with custom layers    |      0.83      |       0.81      |       0.83
|           EfficientNet          |      0.88      |       0.88      |       0.84
| EfficientNet with custom layers |      0.82      |       0.8       |       0.79
+---------------------------------+----------------+-----------------+---------------
```

It is observed that there is a significant increase in the F1-Score of all the other models compared to the baseline model.

The Transfer Learning using custom the pre-trained models with or without the custom layers have yielded a consistent results of over 80%.

It should be noted that this was achieved even with an imbalance of data. Though the imbalance of the data was compensated with generating almost equal number of patches for both classes, the number of patches generated were only a few thousands. Due to system limitations, these models could not trained with huge data. Inspite of this, the models have performed well giving results of over 80%.