**IMAGE RECOGNITION WITH IBM VISUAL RECOGNITION**

**PHASE-4**

**Context**

A popular component of computer vision and deep learning revolves around identifying faces for various applications from logging into your phone with your face or searching through surveillance images for a particular suspect. This dataset is great for training and testing models for face detection, particularly for recognising facial attributes such as finding people with brown hair, are smiling, or wearing glasses. Images cover large pose variations, background clutter, diverse people, supported by a large quantity of images and rich annotations.

CODING:

import pandas as pd

import numpy as np

import cv2

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import f1\_score

from keras.applications.inception\_v3 import InceptionV3, preprocess\_input

from keras import optimizers

from keras.models import Sequential, Model

from keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D

from keras.callbacks import ModelCheckpoint

from keras.preprocessing.image import ImageDataGenerator, array\_to\_img, img\_to\_array, load\_img

from keras.utils import np\_utils

from keras.optimizers import SGD

from IPython.core.display import display, HTML

from PIL import Image

from io import BytesIO

import base64

import tensorflow as tf

plt.style.use('ggplot')

%matplotlib inline

main\_folder = '../input/celeba-dataset/'

images\_folder = main\_folder + 'img\_align\_celeba/img\_align\_celeba/'

EXAMPLE\_PIC = images\_folder + '000506.jpg'

TRAINING\_SAMPLES = 10000

VALIDATION\_SAMPLES = 2000

TEST\_SAMPLES = 2000

IMG\_WIDTH = 178

IMG\_HEIGHT = 218

BATCH\_SIZE = 16

NUM\_EPOCHS = 20

df\_attr = pd.read\_csv(main\_folder + 'list\_attr\_celeba.csv')

df\_attr.set\_index('image\_id', inplace=True)

df\_attr.replace(to\_replace=-1, value=0, inplace=True) #replace -1 by 0

df\_attr.shape

# plot picture and attributes

img = load\_img(EXAMPLE\_PIC)

plt.grid(False)

plt.imshow(img)

df\_attr.loc[EXAMPLE\_PIC.split('/')[-1]][['Smiling','Male','Young']]

df\_partition = pd.read\_csv(main\_folder + 'list\_eval\_partition.csv')

df\_partition.head()

df\_partition['partition'].value\_counts().sort\_index()

df\_partition.set\_index('image\_id', inplace=True)

df\_par\_attr = df\_partition.join(df\_attr['Male'], how='inner')

df\_par\_attr.head()

def load\_reshape\_img(fname):

img = load\_img(fname)

x = img\_to\_array(img)/255.

x = x.reshape((1,) + x.shape)

return x

def generate\_df(partition, attr, num\_samples):

'''

partition

0 -> train

1 -> validation

2 -> test

'''

df\_ = df\_par\_attr[(df\_par\_attr['partition'] == partition)

& (df\_par\_attr[attr] == 0)].sample(int(num\_samples/2))

df\_ = pd.concat([df\_,

df\_par\_attr[(df\_par\_attr['partition'] == partition)

& (df\_par\_attr[attr] == 1)].sample(int(num\_samples/2))])

# for Train and Validation

if partition != 2:

x\_ = np.array([load\_reshape\_img(images\_folder + fname) for fname in df\_.index])

x\_ = x\_.reshape(x\_.shape[0], 218, 178, 3)

y\_ = np\_utils.to\_categorical(df\_[attr],2)

# for Test

else:

x\_ = []

y\_ = []

for index, target in df\_.iterrows():

im = cv2.imread(images\_folder + index)

im = cv2.resize(cv2.cvtColor(im, cv2.COLOR\_BGR2RGB), (IMG\_WIDTH, IMG\_HEIGHT)).astype(np.float32) / 255.0

im = np.expand\_dims(im, axis =0)

x\_.append(im)

y\_.append(target[attr])

return x\_, y\_

# Train data

x\_train, y\_train = generate\_df(0, 'Male', TRAINING\_SAMPLES)

# Train - Data Preparation - Data Augmentation with generators

train\_datagen = ImageDataGenerator(

preprocessing\_function=preprocess\_input,

rotation\_range=30,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

)

train\_datagen.fit(x\_train)

train\_generator = train\_datagen.flow(

x\_train, y\_train,

batch\_size=BATCH\_SIZE,

)

# Validation Data

x\_valid, y\_valid = generate\_df(1, 'Male', VALIDATION\_SAMPLES)

# Import InceptionV3 Model

inc\_model = InceptionV3(weights='../input/inceptionv3/inception\_v3\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5',

include\_top=False,

input\_shape=(IMG\_HEIGHT, IMG\_WIDTH, 3))

print("number of layers:", len(inc\_model.layers))

#inc\_model.summary()

#Adding custom Layers

x = inc\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation="relu")(x)

x = Dropout(0.5)(x)

x = Dense(512, activation="relu")(x)

predictions = Dense(2, activation="softmax")(x)

# creating the final model

model\_ = Model(inputs=inc\_model.input, outputs=predictions)

# Lock initial layers to do not be trained

for layer in model\_.layers[:52]:

layer.trainable = False

# compile the model

model\_.compile(optimizer=SGD(lr=0.0001, momentum=0.9)

, loss='categorical\_crossentropy'

, metrics=['accuracy'])

checkpointer = ModelCheckpoint(filepath='weights.best.inc.male.hdf5',

verbose=1, save\_best\_only=True)

hist = model\_.fit\_generator(train\_generator

, validation\_data = (x\_valid, y\_valid)

, steps\_per\_epoch= TRAINING\_SAMPLES/BATCH\_SIZE

, epochs= NUM\_EPOCHS

, callbacks=[checkpointer]

, verbose=1

)

**Conclusion:**

The built model using transfer learning from the InceptionV3 and adding custom layers successfully recognize the gender giving certain picture with 94.8% of accuracy over the test data. Nevertheless, there are some limitations detected and opportunities for improvements: