**ASSIGNMENT-3**

import numpy as np import pandas as pd

from PIL import ImageFile from tqdm import tqdm import h5py import cv2

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from sklearn.model\_selection import train\_test\_split from sklearn.metrics import confusion\_matrix from sklearn.metrics import plot\_confusion\_matrix

from tensorflow.keras.utils import to\_categorical from tensorflow.keras.preprocessing import image as keras\_image from tensorflow.keras.models import Sequential, load\_model from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Activation, Dropout from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint from tensorflow.keras.layers import LeakyReLU def model():

model = Sequential()

model.add(Conv2D(128, (3, 3), input\_shape=x\_train.shape[1:])) model.add(LeakyReLU(alpha=0.02))

model.add(MaxPooling2D(pool\_size=(2, 2))) model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3))) model.add(LeakyReLU(alpha=0.02))

model.add(MaxPooling2D(pool\_size=(2, 2))) model.add(Dropout(0.25))

model.add(GlobalMaxPooling2D())

model.add(Dense(512)) model.add(LeakyReLU(alpha=0.02)) model.add(Dropout(0.5)) model.add(Dense(10)) model.add(Activation('softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

model = model()

# To save the best model checkpointer = ModelCheckpoint(filepath='weights.best.model.hdf5', verbose=2, save\_best\_only=True)

# To reduce learning rate dynamically lr\_reduction = ReduceLROnPlateau(monitor='val\_loss', patience=5, verbose=2, factor=0.2)

# Train the model history = model.fit(x\_train, y\_train, epochs=75, batch\_size=32, verbose=2,

validation\_data=(x\_valid, y\_valid), callbacks=[checkpointer,

data\_generator = keras\_image.ImageDataGenerator(shear\_range=0.3,

zoom\_range=0.3, rotation\_range=30, horizontal\_flip=True)

dg\_history = model.fit\_generator(data\_generator.flow(x\_train, y\_train, batch\_size=64), steps\_per\_epoch = len(x\_train)//64, epochs=7, verbose=2, validation\_data=(x\_valid, y\_valid), callbacks=[checkpointer,lr\_reduction])