NAME :M SUNITHA

YEAR : FINAL CSE

COLLEGE: MAHENDRA ENGINEERING COLLEGE FOR WOMEN

SUB : IBM

import bumpy as NP

import pandas as pd

from PIL import Image File

from TQM import TQM

import h5py

import cv2

import mat plot lib.plot as pl

import seaborne as ans

from learn.model\_selection import train\_test\_split

from learn.metrics import confusion\_matrix

from learn.metrics import plot\_confusion\_matrix

from ten so flow.k eras.u til import to\_categorical

from ten so flow.k eras .processing import image as k eras\_image

from ten so flow.k eras.models import Sequential, load\_model

from ten so flow.k eras.layers import Dense

from ten so flow.k eras.layers import Activation, Dropout

from ten so flow. K eras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D

from ten so flow.k eras.callbacks import Reduce LR On Plateau, Model Checkpoint

from ten so flow.k eras.layers import Leaky Re LU

def model():

model = Sequential()

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

model.add(Leaky Re LU(alpha=0.02))

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

model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3)))

model.add(Leaky Re LU(alpha=0.02))

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

model.add(Dropout(0.25))

model.add(GlobalMaxPooling2D())

model.add(Dense(512))

model.add(Leaky Re LU(alpha=0.02))

model.add(Dropout(0.5))

model.add(Dense(10))

model.add(Activation('softball'))

model.compile(loss='categorical\_cross curter optimizer=madam', metrics=['accuracy'])

return model

model = model()

# To save the best model

Check pointer = Model Check point (file path='weights.best.model.hdf5', verbose=2, save\_best\_only=True)

# To reduce learning rate dynamically

Lr\_reduction = Reduce LRO n Plateau(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=[cheek pointer,

data\_generator = k eras\_image.Image Data Generator(shear\_range=0.3,

zoom\_range=0.3,

rotation\_range=30,

horizontal\_flip=True)

dag\_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=[check pointer,is\_reduction])