# import the required packages

import numpy as np

import argparse

import imutils

import sys

import cv2

# we will pass argument using argument parser so construct argument parser.

argv = argparse.ArgumentParser()

argv.add\_argument("-m", "--model", required=True,

help="specify path to pre-trained model")

argv.add\_argument("-c", "--classes", required=True,

help="specify path to class labels file")

argv.add\_argument("-i", "--input", ead()

# to exit video stream

if not grabbed:

print("[INFO] No frame read from the stream - Exiting...")

sys.exit(0)

# or else it read

originals.append(frame) # save

frame = imutils.resize(frame, width=400)

frames.append(frame)

# frames array is filled we can construct our blob

blob = cv2.dnn.blobFromImages(frames, 1.0,

(SAMPLE\_SIZE, SAMPLE\_SIZE),

(114.7748, 107.7354, 99.4750),

swapRB=True, crop=True)

blob = np.transpose(blob, (1, 0, 2, 3))

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

# Predict activity using blob

gp.setInput(blob)

outputs = gp.forward()

label = ACT[np.argmax(outputs)]

# for adding lables

for frame in originals:

# append predicted activity

cv2.rectangle(frame, (0, 0), (300, 40),

(0, 0, 0), -1)

cv2.putText(frame, label, (10, 25),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.8,

(255, 255, 255), 2)

# if displayed is yes

if args["display"] > 0:

cv2.imshow("Activity Recognition", frame)

key = cv2.waitKey(1) & 0xFF

# to exit

if key == ord("q"):

break

# for output video being already given

# initialise the witer variable

if args["output"] != "" and writer is None:

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v') # \*'MJPG' for .avi format

w type=str, default="",

help="specify path to video file")

argv.add\_argument("-o", "--output", type=str, default="",

help="path to output video file")

argv.add\_argument("-d", "--display", type=int, default=1,

help="to display output frame or not")

argv.add\_argument("-g", "--gpu", type=int, default=0,

help="whether or not it should use GPU")

args = vars(argv.parse\_args())

# declare an variable to open and load contents of labels of activity .

# specify size here for the frames.

ACT = open(args["classes"]).read().strip().split("\n")

SAMPLE\_DURATION = 16

SAMPLE\_SIZE = 112

# Load the Deep Learning model.

print("Loading The Deep Learning Model For Human Activity Recognition")

gp = cv2.dnn.readNet(args["model"])

# Check if GPU will be used here

if args["gpu"] > 0:

print("setting preferable backend and target to CUDA...")

gp.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

gp.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

# Grab the pointer to the input video stream

print(" Accessing the video stream...")

vs = cv2.VideoCapture(args["input"] if args["input"] else 0)

writer = None

fps = vs.get(cv2.CAP\_PROP\_FPS)

print("Original FPS:", fps)

# Detect continoulsy till terminal is expilicitly closed

while True:

# Frame intilasation

frames = [] # frames for processing

originals = [] # original frames

# Use sample frames

for i in range(0, SAMPLE\_DURATION):

# Read a frame from the video stream

(grabbed, frame) = vs.r riter = cv2.VideoWriter(args["output"], fourcc, fps,

(frame.shape[1], frame.shape[0]), True)

# write frame to output

if writer is not None:

writer.write(frame)