**Implementing Real-time Object Detection System using PyTorch and OpenCV**

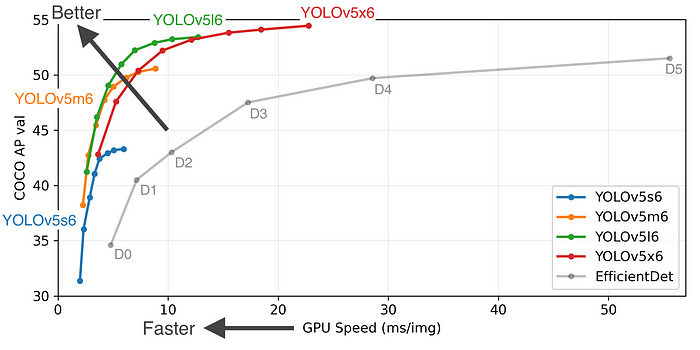
Olvasás a YourTube-ról

A prototípushoz nem biztos, hogy új videót kell készítenie, hanem használjon egyet a sok online elérhető videó közül. Ilyen esetben beolvashatja a videófolyamot a youtube-ról.

import cv2 # opencv2 package for python.  
import pafy # pafy allows us to read videos from youtube.URL = "https://www.youtube.com/watch?v=dQw4w9WgXcQ" #URL to parse  
play = pafy.new(self.\_URL).streams[-1] #'-1' means read the lowest quality of video.  
assert play is not None # we want to make sure their is a input to read.  
stream = cv2.VideoCapture(play.url) #create a opencv video stream.

A modell betöltése

Az egyszerűség kedvéért mi a YoloV5-t választjuk, mivel gyors következtetéseket biztosít, ami kritikus fontosságú a valós idejű alkalmazásunk számára. Más modelleket is megnézhetünk, mint például a FasterRCNN.



A modellt közvetlenül a PyTorch hubból tölthetjük be, a kód első futtatásakor néhány percig tarthat, mivel a modell letöltődik az internetről, de a következő alkalommal már közvetlenül a lemezről töltődik be.

from torch import hub # Hub contains other models like FasterRCNNmodel = torch.hub.load( \  
 'ultralytics/yolov5', \  
 'yolov5s', \  
 pretrained=True)

Egyetlen képkocka pontozása

A következtetés elvégzéséhez használt eszköz óriási különbséget jelent a következtetés sebességében, a modern mélytanulási modellek akkor működnek a legjobban, ha GPU-kkal dolgozunk, így ha van egy CUDA kernellel rendelkező GPU-nk, az hatalmas mértékben javítja a teljesítményt. Tapasztalataim szerint a rendszer akár egyetlen GPU-val is képes 45-60 képkockát elérni másodpercenként, míg egy CPU legfeljebb 25-30 képkockát adhat.

"""  
The function below identifies the device which is availabe to make the prediction and uses it to load and infer the frame. Once it has results it will extract the labels and cordinates(Along with scores) for each object detected in the frame.  
"""def score\_frame(frame, model):  
 device = 'cuda' if torch.cuda.is\_available() else 'cpu'  
 model.to(device)  
 frame = [torch.tensor(frame)]  
 results = self.model(frame)  
 labels = results.xyxyn[0][:, -1].numpy()  
 cord = results.xyxyn[0][:, :-1].numpy()  
 return labels, cord

Dobozok kirajzolása a keret fölé

Miután kiértékeltük a keretet, az azonosított objektumokat és dobozaikat a keret fölé kell rajzolnunk, mielőtt a keretet a kimeneti folyamba írnánk. Ehhez használhatjuk az OpenCV képfeldolgozó eszközkészletét.

"""  
The function below takes the results and the frame as input and plots boxes over all the objects which have a score higer than our threshold.  
"""  
def plot\_boxes(self, results, frame):  
labels, cord = results  
n = len(labels)  
x\_shape, y\_shape = frame.shape[1], frame.shape[0]  
for i in range(n):  
row = cord[i]  
# If score is less than 0.2 we avoid making a prediction.  
if row[4] < 0.2:   
continue  
x1 = int(row[0]\*x\_shape)  
y1 = int(row[1]\*y\_shape)  
x2 = int(row[2]\*x\_shape)  
y2 = int(row[3]\*y\_shape)  
bgr = (0, 255, 0) # color of the box  
classes = self.model.names # Get the name of label index  
label\_font = cv2.FONT\_HERSHEY\_SIMPLEX #Font for the label.  
cv2.rectangle(frame, \  
(x1, y1), (x2, y2), \  
bgr, 2) #Plot the boxes  
cv2.putText(frame,\  
classes[labels[i]], \  
(x1, y1), \  
label\_font, 0.9, bgr, 2) #Put a label over box.  
return frame

Hozzunk létre egy videó streaming bemenetet.

Töltsük be a modellt.

Amíg a bemenet elérhető, olvassuk be a következő képkockát.

Pontozzuk a képkockát, hogy megkapjuk a címkéket és a koordinátákat.

Rajzolja a dobozokat az észlelt objektumok fölé.

Írja a feldolgozott képkockát a kimeneti videófolyamra.

Hat egyszerű lépés a működéshez, bár hozzáadunk néhány infrastrukturális kódot, hogy az alkalmazást robusztusabbá tegyük, az alapok ugyanazok. Tehát csináljuk meg.

"""  
The Function below oracestrates the entire operation and performs the real-time parsing for video stream.  
"""  
def \_\_call\_\_(self):  
player = self.get\_video\_stream() #Get your video stream.  
assert player.isOpened() # Make sure that their is a stream.   
#Below code creates a new video writer object to write our  
#output stream.  
x\_shape = int(player.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
y\_shape = int(player.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
four\_cc = cv2.VideoWriter\_fourcc(\*"MJPG") #Using MJPEG codex  
out = cv2.VideoWriter(out\_file, four\_cc, 20, \  
(x\_shape, y\_shape))   
ret, frame = player.read() # Read the first frame.  
while rect: # Run until stream is out of frames  
start\_time = time() # We would like to measure the FPS.  
results = self.score\_frame(frame) # Score the Frame  
frame = self.plot\_boxes(results, frame) # Plot the boxes.  
end\_time = time()  
fps = 1/np.round(end\_time - start\_time, 3) #Measure the FPS.  
print(f"Frames Per Second : {fps}")  
out.write(frame) # Write the frame onto the output.  
ret, frame = player.read() # Read next frame.

Vegyük ezeket az összetevőket, és csomagoljuk be őket egy szép osztályba, amely meghívható az URL-lel és a kimeneti fájllal együtt, amelyre a kimeneti adatfolyamot írni kívánja.

Forrás

Akash Agnihotri-Implementing Real-time Object Detection System using PyTorch and OpenCV (https://towardsdatascience.com/implementing-real-time-object-detection-system-using-pytorch-and-opencv-70bac41148f7)