Test.py → Classes2Image.py

model.to(device) 'cuda' 'cpu'
modelin çalışacağı platformun seçimi

frame = [frame] ← olmasada çalışıyor ?

tespit işleminin yapılacağı resimin listelenmesi yani 3 boyutlu bir matris haline gelmesi

results = model(frame)
# oluşturulan matrisin modele sokularak çıktı alınması

labels, cord = results....
labels, tespit edilen her objenin label indeksini barındıran liste.
cord, tespit edilen her objenin bounding box koordinatlarını içeren liste

n = len(labels) or n = len(cord)
# kaç objenin tespit edildiği saptanır. labels ve cord ~~elemanları~~ değişkenleri tespit edilen her eleman için değer tutan listelerdir. Yani uzunlukları tespit edilen eleman sayısına eşittir.

.... döngü i ← in range(n)
row = cord[i]
# i indexli obje için (row: [0] num1 x1, [1] num2 y1, [2] num3 x2, [3] num4 y2, [4] num5 predict) elemanlarını tutan değişken

if row[4] >= 0.7: → Eğer tespit edilen objenin:
1- doğruluk oranı
2- obje olma oranı
> hangisi bilmiyorum ?
'0.70' in üstü ise çizim yap

Test 3 → Extra Modules.ipynb

results.pandas().xyxy[0] ← jupyter notebook için

results içerisinde saklanan değerleri bir tablo yapar.
Tablonun içeriği : xmin, ymin, xmax, ymax, confidence, class, name
num1 num2 num3 num4 num5
bundan da önceki notta bakşedilen num1, num2, ..., num5 parametrelerinin neyi temsil ettiğini anlamış olduk

results.print()

resimde tespit edilen sonuçları yazdırır. Resimin boyutu, tespit edilen objeler ve adetleri. Ekstra olarak süre, vs. gibi sonuçları da yazdırır.

results.save()

tespit sonuçlarını kendi oluşturduğu (oluşturacağı) klasöre kayıt eder. RGB formatı resimler ile çalışır bu sebeple BGR resimin çıktısında renkler farklı görünür.

Modelin Çalışacağı Cihaz Seçimi:

model.cpu()
model.cuda() → "cpu" or "cuda"
model.to(device)

Çıktı\Çıkarım Ayarları

#model.conf = ⟶ NMS confidence threshold ← non maximum suppression → aynı obje üzerine birden fazla bounding box çizmemek için
.iou = ⟶ NMS IoU threshold
" .agnostic = ⟶ NMS class-agnostic
" .multi_label = ⟶ NMS multiple labels per box
" .classes = ⟶ filter by classes, =[0,15,16] means only persons, cats and dogs
" .max_det = ⟶ maximum number of detections per image
" .amp = ⟶ amp inference (automatic mixed precision)

Model Yükleme

model = torch.hub.load(<link>, <model>) ← default yolov5 modellerini çağırmak
model = torch.hub.load(<link>, <model>, path="") ← kendi modelinizi içe aktarma
custom → <model>
your model → path

Model içe aktarma sırasında

```
Model = torch.hub.load("ultralytics/yolov5", "yolov5s", +)
```

+ device = 'cpu'
# modeli cpu üzerinden çalıştırır

+ _verbose = False
# sessizce yükleme

+ channels = 4 (default)
# daha önceden eğitilmiş bir modelin 3kanal yerine 4 kanal ile yüklenmesini sağlar.
Böylece önceden eğitilmiş girdi katmanıyla aynı şekle sahip olmayan ilk girdi katmanı haricinde önceden eğitilmiş ağırlıklardan oluşacaktır.

+ classes = ...
# 80 sınıf yerine belirtilen adet kadar sınıf için aktarılacaktır.

+ force_reload = True
# Yukarıdaki adımlar ile ilgili sorun yaşanır ise en son YoloV5 sürümünü (ve ayarlarını) zorla kurar

Multi GPU

* YoloV5 aynı GPU'ya yüklenim threadler aracılığı ile eş zamanlı çalıştırılabilir. ← Text3

Pytorch.hub sitesinden de bakılabilir

JSON Results

* results.pandas().xyxy[0].to_json(orient="records")
# sonuçları bir json dosyasına kayıt eder