Computer vision

Homework 12

Parsa Eissazadeh 97412364

سوال 1

Overfitting : زمانی پیش می آِید که شبکه از حدی قوی تر شده است . این قوی تر شدن علل مختلفی میتواند داشته باشد .

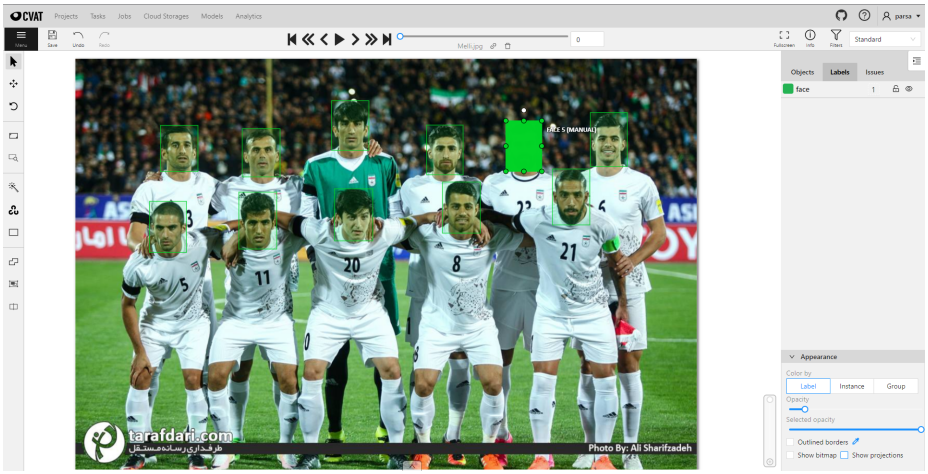
زیاد بودن تعداد داده ها یکی از علل overfit است که اگر نسبت به تعداد نورون های شبکه از حدی بیشتر باشد ، شبکه به جای اینکه الگو ها را پیدا بکند همانند دیکشنری ای می شود از داده های ورودی به جدولشان .

نتیجه ای که overfitting میتواند داشته باشد این است که در هنگام train کردن به دقت 100 درصد میرسیم و به هنگام validation دقت پایینی gain میکنیم .

Underfitting :

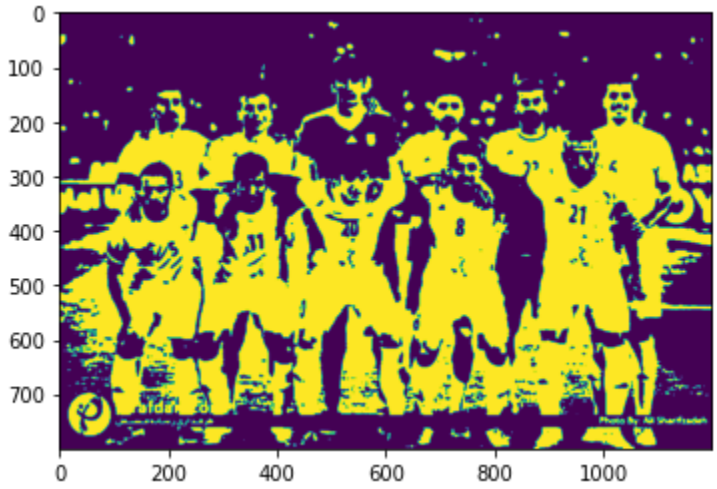
سوال 2

در ابتدا توسط ابزار آنلاین Cvat ، برچسب گذاری رو انجام میدیم :

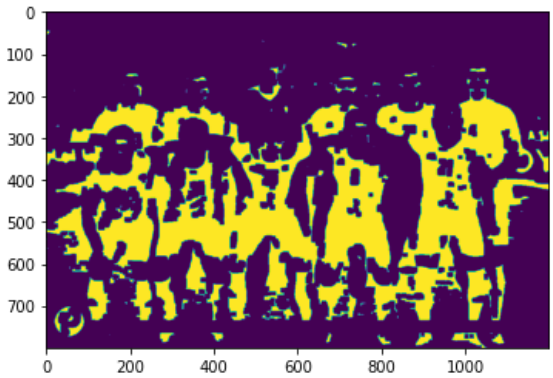


در ابتدا (قبل از اعمال الگوریتم پنجره لغزان) سعی میکنیم اندکی دامنه پنجره ها را کمتر کنیم .

: Thresholding

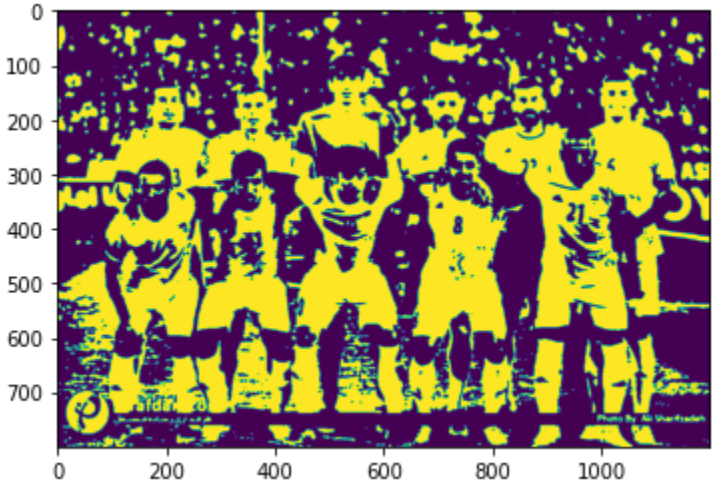


حال عکس را erode میکنیم تا جزییات پست سر بازیکن ها حذف شود :

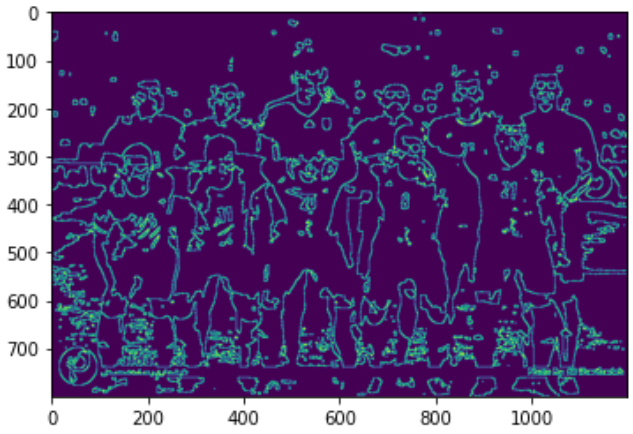


جزییات پشت سر به خوبی حذف شدند و تنها بازیکنان باقی ماندند اما صورت بعضی از بازیکنان نیز از دست رفت که مطلوب ما **نیست** .

صورت بازیکن های جلویی تقریبا از بین رفته اند زیرا همانند بازیکن های عقبی در نور نیستند . در نتیجه از کلاهه استفاده میکنیم :



پس از استفاده از کلاهه ، جزییات پشت سر بازیکنان پررنگ تر شدند . همچنین بعد از اعمال لبه یاب به الگوریتم زیر رسیدیم که باز هم بین صورت بازیکنان و بک گراند تفاوتی ایجاد نمی کند :



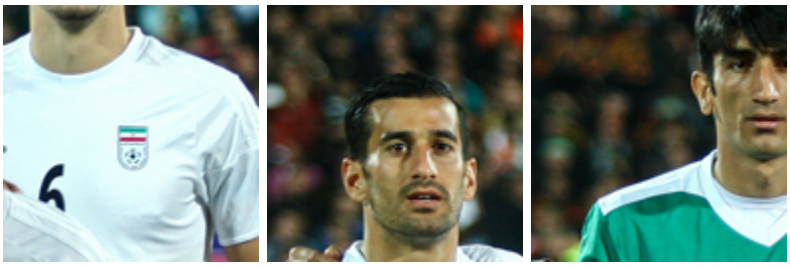
در نتیجه تنها سراغ روش های عمیق باید رفت برای حل این مساله .

پنجره لغزان

برای پیدا کردن پنجره های لغزان از لینک زیر کمک گرفتم :

https://pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/

به کمک این لینک عکس های ناحیه های پیشنهادی استخراج شدند :



- سایز پنجره ها 128 در 128 است .

پیدا کردن annotation ها

مدل های مختلفی برای export کردن پروژه از Cvat وجود دارد که من مدل Cvat for images را انتخاب کردم زیرا مختصات قطر ها را میداد .

مختصات قطر ها در یک فایل xml داده شده است در نتیجه نیاز داریم آن را parse بکنیم .

```
import xml.etree.ElementTree as ET

tree = ET.parse("/content/annotations.xml")

root = tree.getroot()

faces = []

# iterate news items
for item in root.findall('./image/box'):
  faces.append(
      (
        int(float(item.attrib['ytl'])),
        int(float(item.attrib['xtl'])),
        int(float(item.attrib['ybr'])),
        int(float(item.attrib['xbr']))
      )
  )
len(faces)
```

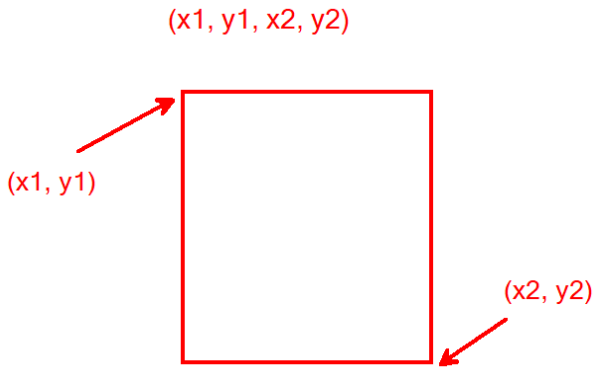
Parse میکنیم و سپس آرایه faces را میسازیم .

محاسبه IoU

از لینک زیر کمک گرفتم :

https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

در ابتدا همه خروجی های مرحله قبل به tuple های چهارتایی از مختصات دو قطر مقابل هم تبدیل کردم :



سپس توسط همین دو قطر مساحت هر مستطیل و ناحیه مشترک بینشان را حساب میکنیم . همچنین داریم :

اجتماع = مساحت مستطیل اول + مساحت مستطیل دوم - مساحت مشترک

سپس به ازای هر ناحیه و هر عکس صورت ، IoU را حساب کردم .

شرطی گذاشتم که اگر IoU از 0.4 بزرگتر بود آن عکس صورت یکی از بازیکنان است . 21 عکس پیدا شد . اما شرطم را که بزرگتر بودن از 0.45 بود را اعمال کردم تنها 4 عکس باقی ماند .

در کد نتیجه هر پنجره و برچسبش را مینویسیم . کد نهایی:

```
result = []

for boxA in suggested :
  for boxB in faces:
    iou = intersection_over_union(boxA, boxB)

    if iou >= .4 :
      region = image[boxA[0]:boxA[2], boxA[1]:boxA[3]]
      time.sleep(0.025)

      result.append((boxA, 'face'))
    else :
      result.append((boxA, 'background'))
```

و به شکل رندم 5 پنجره را انتخاب میکنیم که کلاسشان را نمایش دهیم :

همانطور که مشاهده میشود نتایج به درستی نشان داده شده اند .