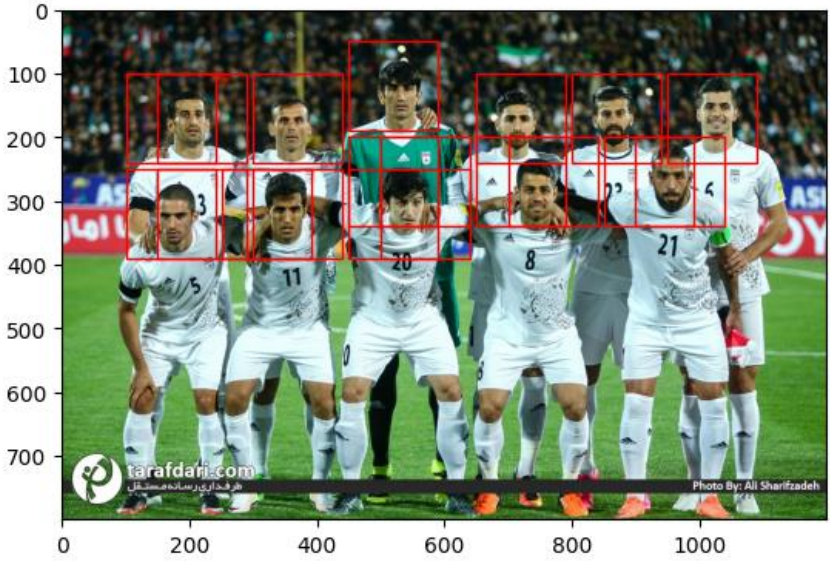
۱. در حالتی که مثل شبکه‌های کم‌عمق و کم‌پارامتر خطای مدل هم برای داده‌های آموزش و هم داده‌های تست زیاد باشد اصطلاحا می‌گوییم مدل Underfit شده است. با افزایش عمق و زیادکردن تعداد پارامترها می‌توانیم مدل را پیچیده‌تر کنیم و توانایی یادگیری شبکه را افزایش دهیم که هرچه بیشتر این کار را ادامه دهیم خطای آموزش را می‌توانیم هر چقدر که بخواهیم پایین بیاوریم ولی در مورد داده‌های تست اینطور نیست که هر چقدر بیشتر جلو رویم و شبکه را پیچیده‌تر کنیم دقت بالاتر برود. تا جایی این اتفاق میفتد اما از یک جایی به بعد اتفاقا هر چقدر تعداد پارامترها را بیشتر کنیم دقت مدل برای داده‌های تست کمتر هم می‌شود. وقتی به محدوده‌ای برسیم که با پیچیده‌کردن مدل در حال کاهش دقت داده‌های تست باشیم (با وجود افزایش دقت داده‌های آموزش) می‌گوییم مدل Overfit شده است. در این محدوده دلیل افزایش دقت داده‌های آموزش حفظ‌کردن این داده‌ها توسط مدل است که ویژگی‌های نامربوطی از داده‌های آموزش را یاد گرفته که قابل تعمیم به داده‌های تست نیستند.

برای اینکه هم مدل Overfit نشود و هم بتوانیم از مزایای شبکه‌های پیچیده عمیق استفاده کنیم می‌توانیم مسئله را برای مدل پیچیده، سخت‌تر کنیم که یک ایده‌ی ساده استفاده از dropout است. یعنی در هر مرحله بصورت تصادفی تعدادی از نورون‌های شبکه را بسوزانیم و دستی مقدار آن‌ها را دستکاری کنیم (مثلا صفر بگذاریم) اگر مدل به اندازه کافی باهوش باشد با استفاده از نورون‌های دیگر باید بتواند به نتیجه درست برسد. که این سختتر کردن مسئله برای مدل می‌تواند باعث جلوگیری از حفظ‌کردن داده‌های آموزش توسط مدل بشود.

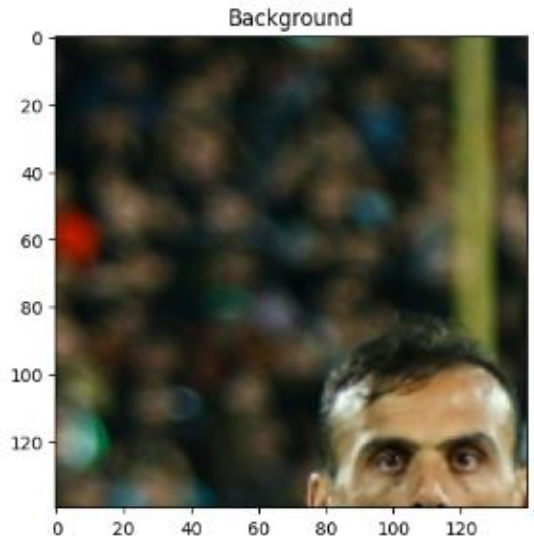
ایده‌ی دیگر استفاده از داده‌افزایی است. قبل از اینکه ورودی را به شبکه بدهیم تغییراتی روی آن اعمال کنیم که ماهیت آن تغییر نکند و به این شکل چندین داده‌ی دیگر از روی یک داده‌ی مشخص تولید کنیم. مثلا در حوزه بینایی کامپیوتر، می‌توانیم تبدیل affine، projective و... روی تصویر ورودی اعمال کنیم بطوری که ماهیت آن را تغییر ندهد. به این شکل dataset ورودی چندین برابر می‌شود و از حفظ کردن اطلاعات بی‌ارزش و نامرتبط تصاویر ورودی توسط شبکه جلوگیری می‌کنیم. انواع مختلفی از داده‌افزایی روی تصاویر وجود دارند:

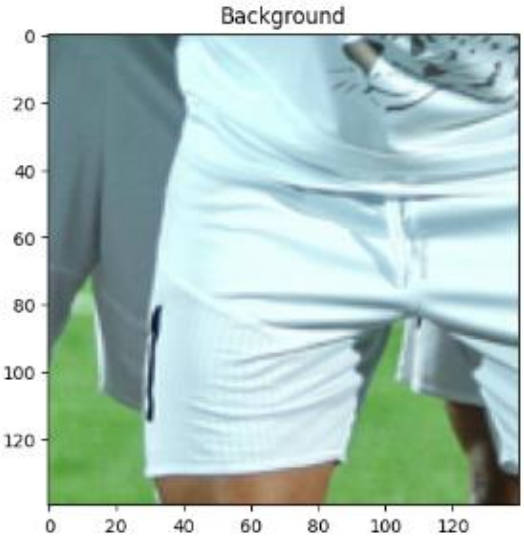
- تبدیلات مختلف هندسی (مثلا flipکردن عمودی و افقی، چرخش با زوایای مختلف، بزرگنمایی، فرمول‌های projective و affine و...)
- تغییر رنگ و نورپردازی
- افزودن نویز (نمک و فلفل، گاوسی، تارکردن و...)

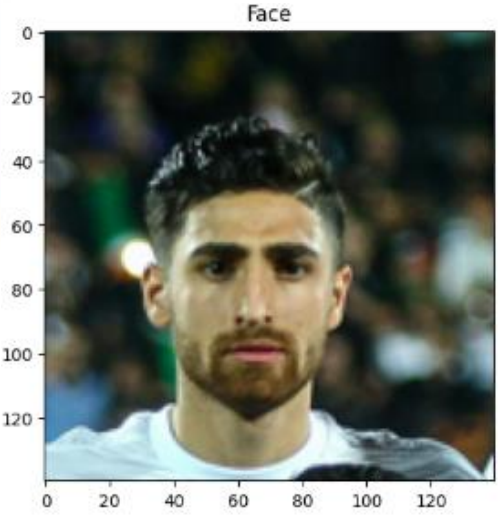


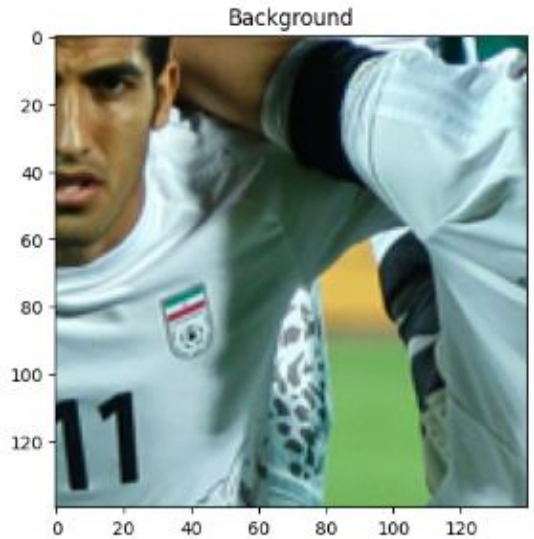
۲. اندازه پنجره لغزان را ۱۴۰x۱۴۰ و با گام ۵۰ پیکسل در نظر گرفتم. حد آستانه‌ی IoU را 0.35 گرفتم.

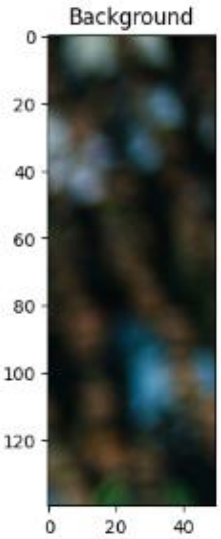
نمایش ۵تا از پنجره‌ها به‌صورت رندوم:





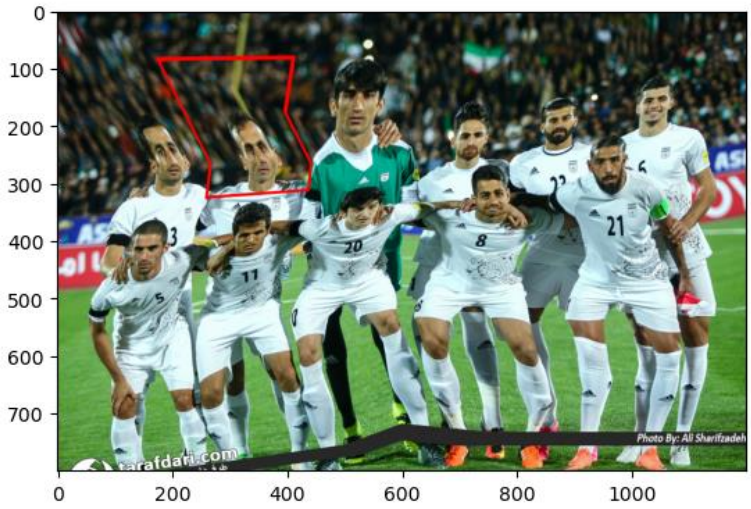






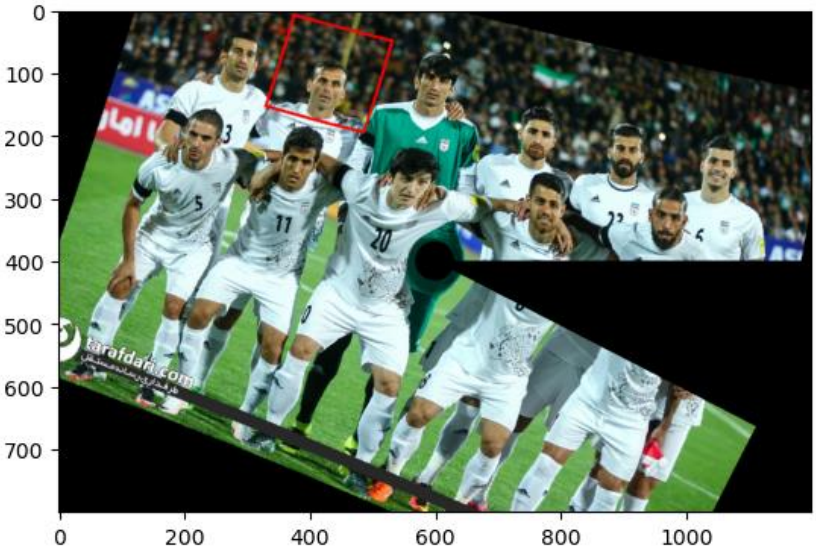
۳. box انتخاب‌شده از تصویر

PiecewiseAffine

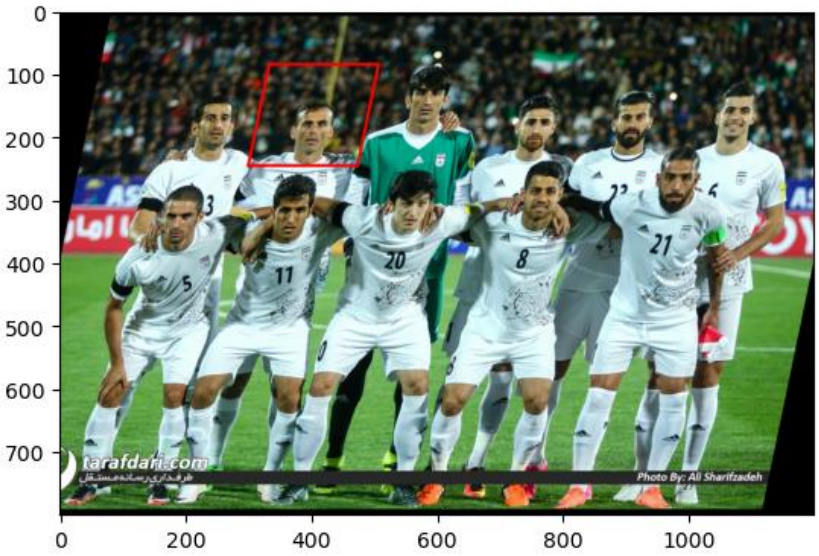


Rotate

WithPolarWarping



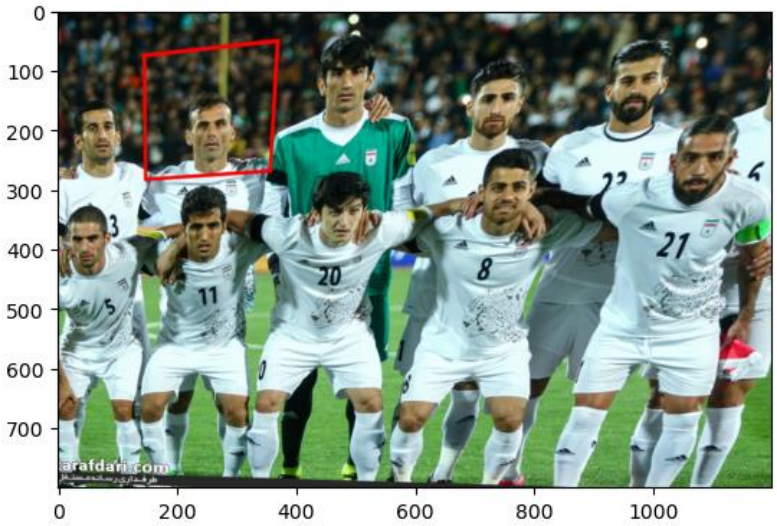
ShearX



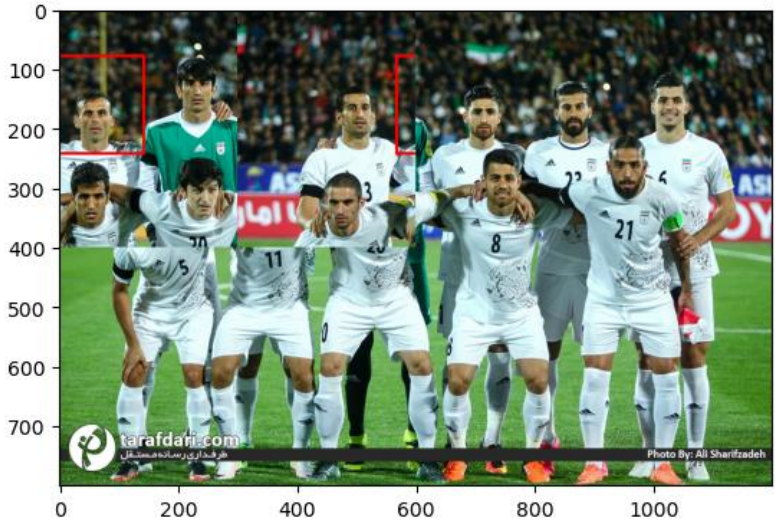
ScaleY

ScaleX

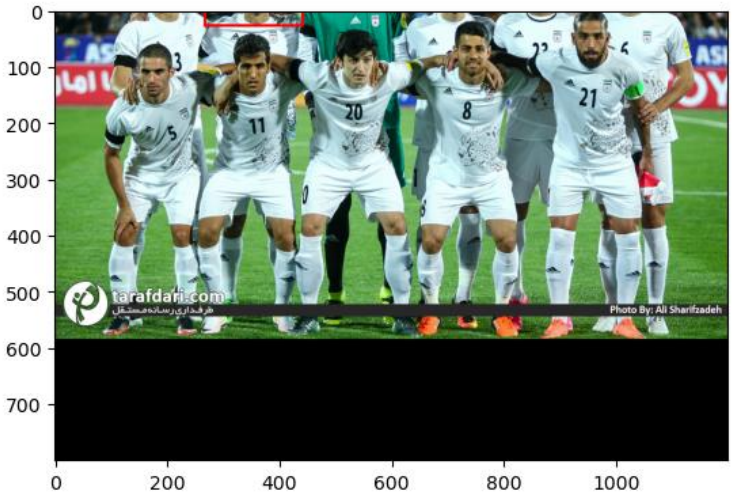
PerspectiveTransform



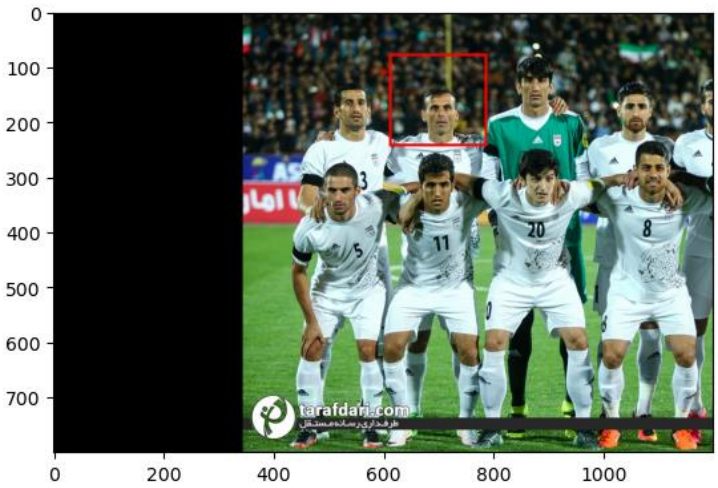
Jigsaw



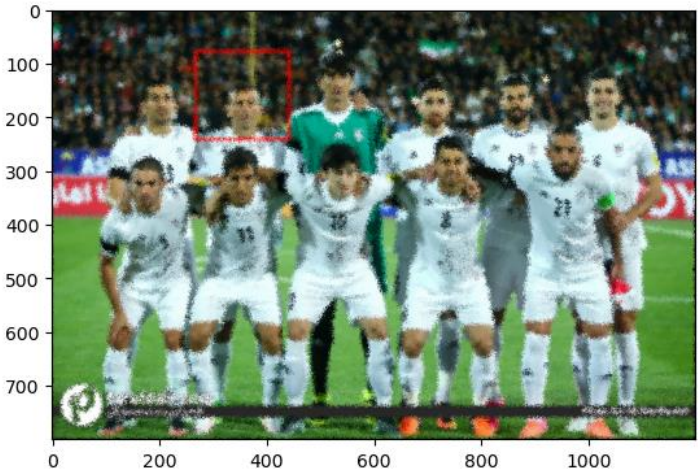
TranslateY



TranslateX



ElasticTransformation



۴.

```
1-what's the effect of padding is equal same? What's another value for padding?
```

مقدار دیگر برای padding ”valid“ است به این معنا که padding اعمال نشود. اگر برای padding مقدار ”same“ را بدهیم به این معناست که طوری padding اعمال شود که سایز تصویر خروجی هم‌اندازه با تصویر ورودی شود. (البته این در صورتی است که strides هم ۱ داده شود وگرنه ابعاد تصویر خروجی کسری از ابعاد تصویر ورودی خواهد شد)

```
2- explain the affection of activation function.
```

تابع فعالساز یا activation function تصمیم می‌گیرد یک نورون باید فعال باشد یا نه و دلیل اصلی استفاده از آن غیرخطی کردن خروجی یک نورون است تا فرآیند آپدیت کردن وزن‌ها در back-propagation ممکن شود و در نتیجه یادگیری برای مدل اتفاق بیفتد.

```
3- explain the affection of using kernel_initializer in layers.
```

نحوه‌ی مقداردهی اولیه وزن‌های لایه‌ها را مشخص می‌کنند. می‌توان توزیع‌های تصادفی مختلفی را برای وزن‌دهی اولیه لایه‌ها انتخاب کرد. در کد داده شده نوت‌بوک از HeNormal استفاده شده به این معنا که توزیع تصادفی نرمال به مرکز صفر و با انحراف معیار $\sqrt{\frac{2}{fan\_in}}$ استفاده شود.

```
4- explain what's the difference between Conv2DTranspose and Conv2D.
```

Conv2D برای انجام عمل کانولوشن و Conv2DTranspose برای معکوس عمل کانولوشن است. Conv2D ابعاد ورودی را در خروجی کوچک می‌کند و Conv2DTranspose ابعاد آن را در خروجی بزرگ می‌کند. در شبکه‌های کانولوشنی encoder-decoder می‌توانیم از Conv2d در قسمت encoder و از Conv2DTranspose در قسمت decoder استفاده کنیم.

```
5- explain downsample_block, double_conv_block and upsample_block functions.
```

- تابع double_conv_block در ورودی تعداد فیلترها و لایه‌های ساخته‌شده از قبل را می‌گیرد و دو لایه‌ی کانولوشنی می‌سازد که کرنل آن‌ها ۳در۳ است و مقداردهی اولیه بصورت رندوم با توزیع تصادفی به مرکز صفر و انحراف معیار $\sqrt{\frac{2}{fan\_in}}$ است. همچنین برای padding ۲ سطر و ستون به ورودی اضافه می‌شود تا سایز ورودی و خروجی لایه‌ها یکسان باشد و به عنوان تابع فعال‌ساز از relu استفاده شده است.

- تابع downsample_block در قسمت encoding شبکه استفاده می‌شود. این لایه از یک بلوک دوتایی کانولوشنی تشکیل شده که با همان تابع double_conv_block ایجاد می‌شود و سپس یک لایه‌ی max pooling برای کوچک کردن ابعاد خروجی استفاده می‌شود. برای pooling از ناحیه‌های ۲در۲ استفاده شده است. سپس یک لایه‌ی Dropout برای سخت‌تر کردن مسئله جهت جلوگیری از حفظ‌کردن اطلاعات گمراه کننده و overfitting قرار داده شده، بطوری که ۳۰درصد از خروجی pooling دراپ می‌شود و به لایه‌ی بعدی نخواهد رسید تا مدل با استفاده از ۷۰درصد بقیه (که تصادفی انتخاب می‌شود) تصمیم‌گیری کند.
- تابع upsample_block در قسمت decoding شبکه استفاده می‌شود. این لایه از معکوس کانولوشن با کرنل۳در۳ و stride ۲تایی استفاده شده. از padding هم بطوری استفاده شده که ابعاد ورودی و خروجی تفاوت نکند. ویژگی‌های استخراج‌شده در قسمت encoding و در لایه‌ای که سایز با لایه‌ی فعلی برابر بوده را با خروجی معکوس کانولوشن concat می‌کند و در کنار هم می‌گذارد. که علاوه بر ویژگی‌های عمیق بدست آمده از ویژگی‌هایی که قبلا در قسمت encoding بدست‌آمده هم استفاده کند. (به عنوان ایده‌ی اصلی UNET) سپس ۳۰درصد مقادیر در Dropout دراپ می‌شوند و در نهایت بلوکی از دو لایه‌ی کانولوشنی استفاده می‌شود.

```
6- why use an optimizer in learning?
```

برای تغییردادن ویژگی‌های شبکه (وزن‌ها، biasها، نرخ یادگیری) در طول فرآیند یادگیری از یک تابع ریاضی یا الگوریتمی به نام optimizer استفاده می‌شود تا بتوانیم loss را کمینه کنیم و در نتیجه دقت مدل را بالا ببریم.

```
7- why use compile function?
```

جهت پیکربندی(کانفیگ) مدل برای آموزش استفاده می‌شود. در آن تنظیمات لازم مثلا optimizer، loss function و معیار (یا معیارها) را مشخص می‌کنیم.

```
8- why are we select categorical_crossentropy in the loss of function?
```

معمولا در مسائل دسته‌بندی از تابع ضرر crossentropy استفاده می‌شود و در اینجا که هدف کلاس‌بندی پیکسل‌های تصویر به بیش از دو حالت است از categorical_crossentropy استفاده شده. (اگر فقط دو کلاس داشتیم از binary_crossentropy استفاده می‌کردیم ولی الان بیش از دو کلاس داریم)

9 - explain earlystopping function

به کمک EarlyStopping می‌توانیم تعیین کنیم وقتی بهبود یک پارامتر مشخص متوقف شد فرآیند آموزش هم متوقف شود. در اینجا مقدار تابع ضرر به عنوان پارامتر مشخص شده تا هر وقت بهبود تابع ضرر متوقف شد آموزش هم متوقف شود.

10 - explain different between fit and compile functions in Keras

تابع compile فقط برای پیکربندی تنظیمات مربوطه فرآیند آموزش است ولی بعد از فراخوانی compile هیچ آموزشی صورت نمی‌گیرد مگراینکه تابع fit صدا زده شود. پس compile همیشه قبل از fit صدا زده می‌شود تا تنظیمات آموزش را مشخص کند و بعد با صدازده شدن fit فرآیند آموزش انجام می‌شود.

11- explain the difference between batch and epoch

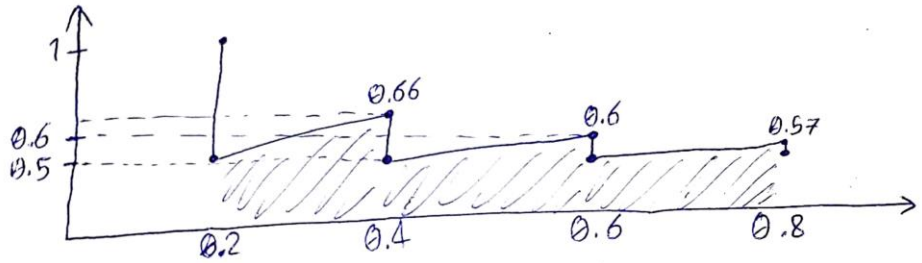
یک batch تعدادی از نمونه‌ها است. بعد از اینکه همه‌ی نمونه‌های یک batch را به شبکه دادیم خروجی‌های شبکه را با خروجی‌های موردانتظار مقایسه می‌کنیم تا وزن‌های شبکه را آپدیت کنیم. ولی epoch ربطی به تعداد نمونه‌ها ندارد. یک epoch یعنی یکبار دادن نمونه‌های dataset به شبکه. به عنوان یک ابرپارامتر مشخص می‌کنیم تعداد بارهایی که dataset به شبکه داده می‌شود چندبار باشد (چند epoch باشد) همچنین تعداد نمونه‌های یک batch هم به عنوان ابرپارامتر قبل از شروع آموزش تنظیم می‌شود.

۵. در ابتدا سطرهای فایل detection را بر اساس مقادیر ستون score بصورت نزولی مرتب کردم.

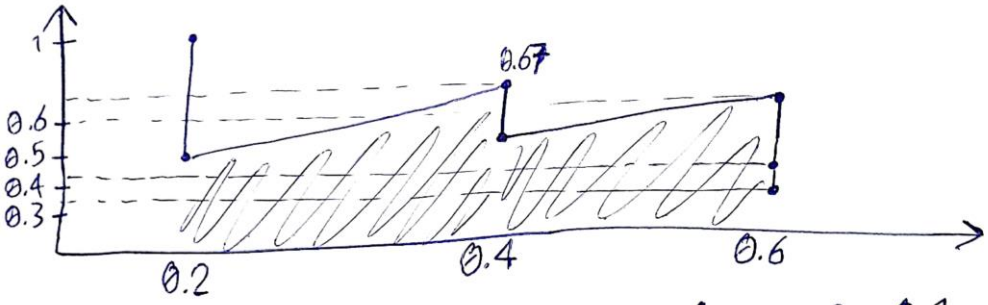
در گام بعدی دو ستون به ازای هر یک از سه حالت IoU ۲۵ و ۵۰ و ۷۵ اضافه کردم که مقادیر Recall و Precision را در آن‌ها ثبت کنم.

x	y	w	h	score	correct_25	Precision	Recall	correct_50	Precision	Recall	correct_75	Precision	Recall
114	31	14	21	0.96	TRUE	1.00	0.20	TRUE	1.00	0.20	FALSE	0.00	0.00
55	72	34	36	0.89	FALSE	0.50	0.20	FALSE	0.50	0.20	FALSE	0.00	0.00
11	5	19	26	0.84	TRUE	0.66	0.40	TRUE	0.67	0.40	TRUE	0.33	0.20
18	39	31	23	0.79	FALSE	0.50	0.40	FALSE	0.50	0.40	FALSE	0.25	0.20
124	136	29	35	0.74	TRUE	0.60	0.60	TRUE	0.60	0.60	TRUE	0.40	0.40
24	98	21	34	0.47	FALSE	0.50	0.60	FALSE	0.50	0.60	FALSE	0.33	0.40
36	150	41	26	0.39	TRUE	0.57	0.80	FALSE	0.43	0.60	FALSE	0.29	0.40
92	153	27	47	0.29	FALSE	0.50	0.80	FALSE	0.38	0.60	FALSE	0.25	0.40

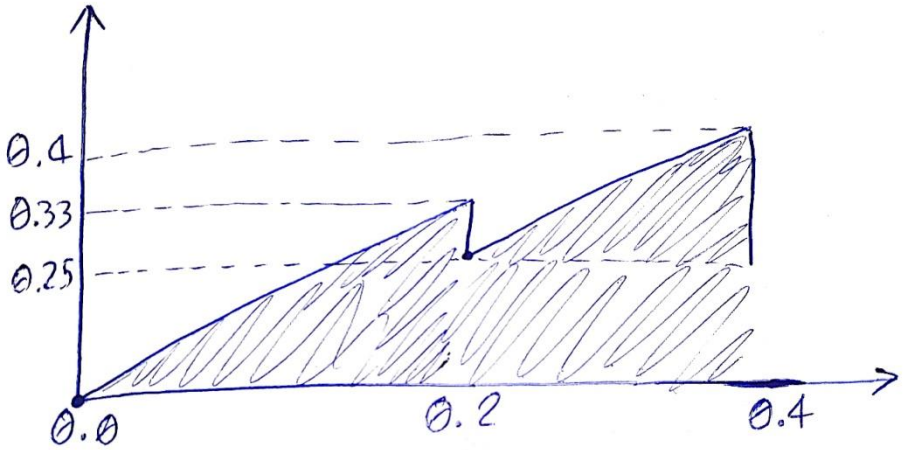
در نهایت در هر سه حالت Precision را برحسب Recall رسم می‌کنیم و سطح زیر نمودار را به عنوان AP بدست می‌آوریم:



$$AP_{25} = 0.6 \times 0.5 + \frac{0.2 \times 0.16}{2} + \frac{0.2 \times 0.1}{2} + \frac{0.2 \times 0.07}{2}$$
$$= 0.3 + 0.016 + 0.01 + 0.007 = 0.323$$



$$AP_{50} = 0.5 \times 0.4 + \frac{0.2 \times 0.17}{2} + \frac{0.2 \times 0.1}{2}$$
$$= 0.2 + 0.017 + 0.01 = 0.227$$



$$AP_{75} = \frac{0.2 \times 0.33}{2} + 0.2 \times 0.25 + \frac{0.2 \times 0.15}{2}$$
$$= 0.033 + 0.05 + 0.015 = 0.098$$

منابع

سوال ۲)

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

https://www.geeksforgeeks.org/how-to-draw-rectangle-on-image-in-matplotlib/

سوال ۳)

https://github.com/aleju/imgaug/issues/716

https://imgaug.readthedocs.io/

https://gist.github.com/Lexie88rus/6b80fbfe94ae85d814aa77d0808fb96b

سوال ۴)

https://www.geeksforgeeks.org/activation-functions-neural-networks/

https://keras.io/api/layers/initializers/

https://stackoverflow.com/questions/68976745/in-keras-what-is-the-difference-between-conv2dtranspose-and-conv2d

https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/