گزارش تمرین دوازدهم بینایی ماشین

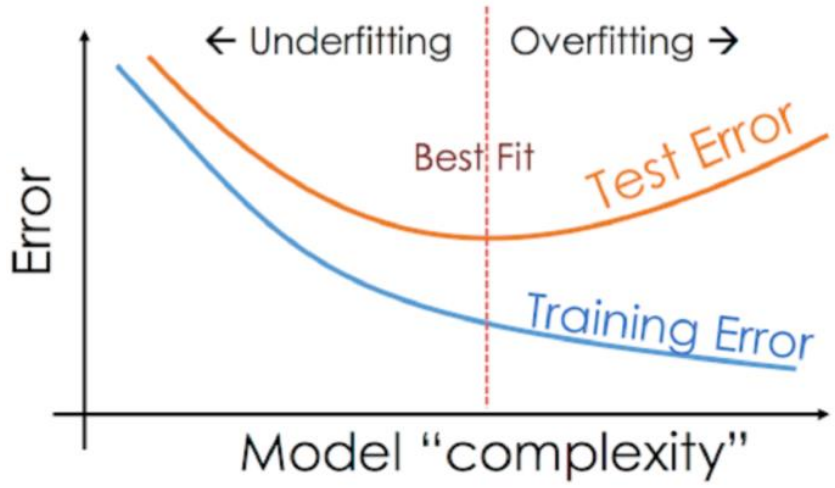
نام تهیه‌کننده: ملیکا نوبختیان

شماره دانشجویی: ۹۷۵۲۲۰۹۴

نسخه: ۱

۱- سوال اول

قبل از صحبت در مورد overfitting و underfitting بهتر است نگاهی به نمودار زیر بیندازیم:

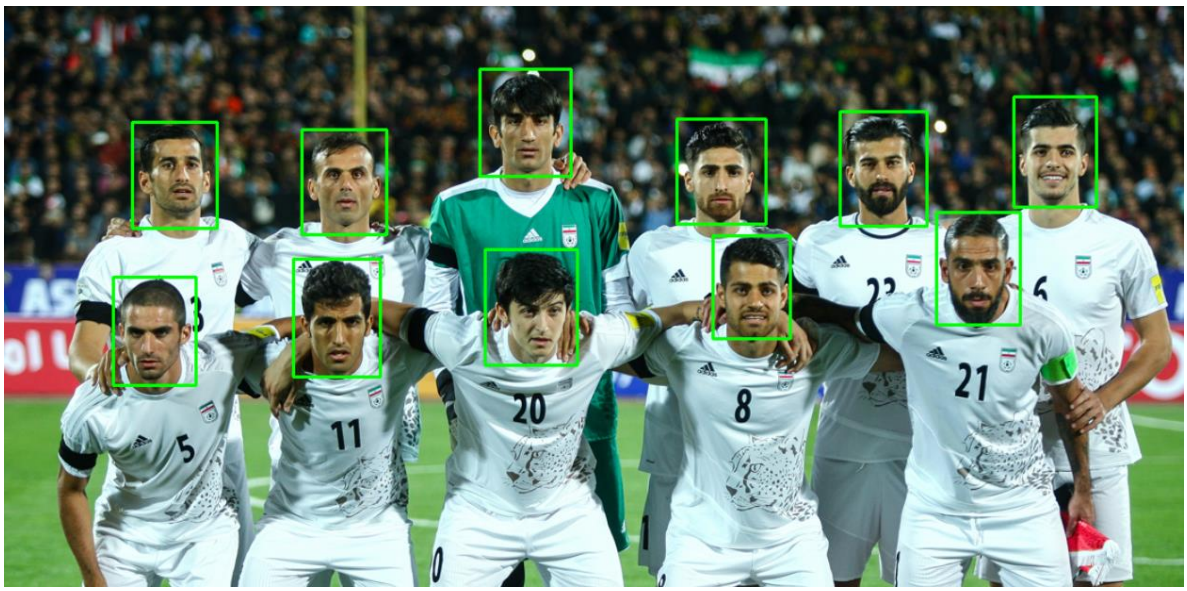


اگر مدل بیش از حد ساده باشد به طوری که نتواند به خوبی ویژگی‌های متمایزکننده کلاس‌های مختلف را یاد بگیرد، underfitting رخ می‌دهد. در این حالت خطا هم روی داده‌های تست هم روی داده‌های آموزشی بالا است. در این حالت افزایش پیچیدگی مدل می‌تواند به ما کمک کند تا این مشکل را حل کنیم.

اما گاهی اوقات پیچیدگی مدل زیاد است و خطا روی داده‌های آموزشی کم است ولی روی داده‌های تست خطا بالا است. در این حالت می‌گوییم overfitting رخ داده‌است. مدل پیچیده آنقدر داده‌های آموزشی را خوب یاد گرفته‌است که با خطای کمی می‌تواند آن‌ها را پیش‌بینی کند اما قدرت تعمیم آن کاهش یافته‌است و روی داده‌های تست به خوبی عمل نمی‌کند. برای حل این مشکل می‌توانیم از data augmentation استفاده کنیم. داده‌افزایی با اضافه کردن داده‌های مختلف شانس اینکه مدل روی یک الگوی نامربوط به برچسب overfit شود را کاهش می‌دهد و قدرت تعمیم بیشتری به مدل می‌دهد. هم چنین استفاده از dropout نیز به ما در حل این مشکل کمک می‌کند.

۲- سوال دوم

در ابتدا با استفاده از ابزار labelme به صورت زیر box هایی که شامل صورت‌های بازیکنان هستند را پیدا کردم:



ما مختصات نقطه بالا سمت چپ و پایین سمت راست box را خواهیم داشت.

برای اینکه sliding window را در تصویر حساب کنیم به شکل زیر عمل می‌کنیم:

```
def sliding_window(image, stepSize, windowSize):
    for y in range(0, image.shape[0], stepSize):
        for x in range(0, image.shape[1], stepSize):
            if y + windowSize[1] < image.shape[0] and x + wi
ndowSize[0] < image.shape[1]:
                yield [[x, y],[x + windowSize[0], y + window
Size[1]]]
```

به این تابع تصویر، اندازه پنجره و اندازه قدمی که با آن در تصویر جلو می‌رویم را می‌رویم را می‌دهیم. هر پنجره‌ای که با این شرایط در محدوده ابعاد تصویر قرار بگیرد را باز خواهیم گرداند.

برای اینکه ابعاد مناسب برای پنجره را تشخیص دهم، ابعاد box هایی که به صورت دستی برای صورت بازیکنان بود را به دست آوردم:

```
1 for box in faces_boxes:
2     width = box[1][0] - box[0][0]
3     height = box[1][1] - box[0][1]
4     print(f'w : {width}, h: {height}')
```

```
w : 78, h: 102
w : 80, h: 99
w : 82, h: 111
w : 86, h: 101
w : 87, h: 109
w : 84, h: 99
w : 73, h: 97
w : 82, h: 108
w : 79, h: 106
w : 78, h: 104
w : 80, h: 100
```

تصمیم گرفتم مقدار max برای هر کدام از width و height را به عنوان ابعاد پنجره در نظر بگیریم که از بیشترین عدد تصویر بالا بزرگ‌تر باشد و مانند box های موجود در تصویر یک مستطیل ایستاده با ابعاد 112 * 90 خواهد بود. اندازه stepSize را نیز برابر ۱۵ قرار دادم.

برای محاسبه IoU به شکل زیر عمل می‌کنیم:

```
def IoU(first_box, second_box):
    #check if they have intersection
    if first_box[1][0] <= second_box[0][0] or second_box[1][0] <=
 first_box[0][0]:
        return 0
    if first_box[0][1] >= second_box[1][1] or first_box[1][1] <=
second_box[0][1]:
        return 0
    xA = max(first_box[0][0], second_box[0][0])
    yA = max(first_box[0][1], second_box[0][1])
    xB = min(first_box[1][0], second_box[1][0])
    yB = min(first_box[1][1], second_box[1][1])

    intersection_area = max(0, xB - xA + 1) * max(0, yB - yA + 1)

    first_box_area = (first_box[1][0] - first_box[0][0] + 1) * (f
irst_box[1][1] - first_box[0][1] + 1)
    second_box_area = (second_box[1][0] - second_box[0][0] + 1) *
 (second_box[1][1] - second_box[0][1] + 1)

    iou = intersection_area / (first_box_area + second_box_area -
 intersection_area)

    return iou
```

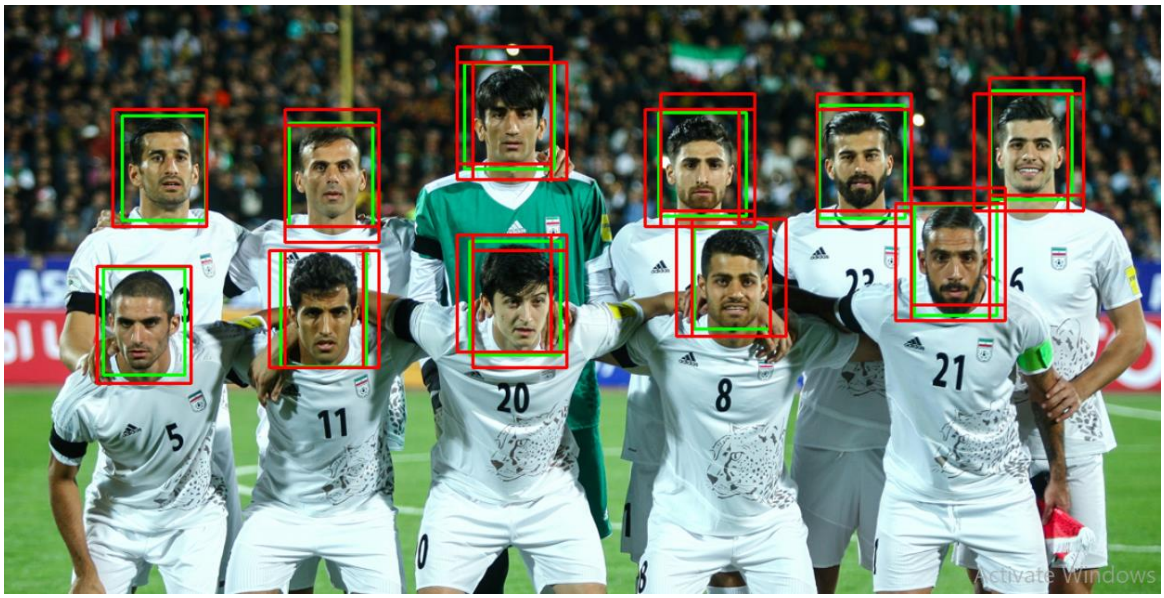
ورودی این تابع مختصات نقاط بالا چپ و پایین راست دو box موردنظر ما خواهد بود. در ابتدا باید ببینیم که آیا دو box موردنظر ما ناحیه مشترکی دارند یا نه. اگر هیچ ناحیه مشترکی نداشته باشند IoU صفر خواهد شد. سپس باید مختصات نقاط مورد نظر ناحیه intersection دو box را محاسبه کنیم. برای نقطه بالا چپ بین x و y هر دو نقطه max می‌گیریم و برای نقطه پایین راست بین x و y این دو نقطه min می‌گیریم. سپس مساحت ناحیه مشترک را محاسبه می‌کنیم و به دنبال آن مساحت box اول و دوم را به دست می‌آوریم. سپس iou را با توجه به مواردی که به دست آوردیم محاسبه می‌کنیم.

حالا به سراغ پیدا کردن proposal های پیشنهادی می‌رویم:

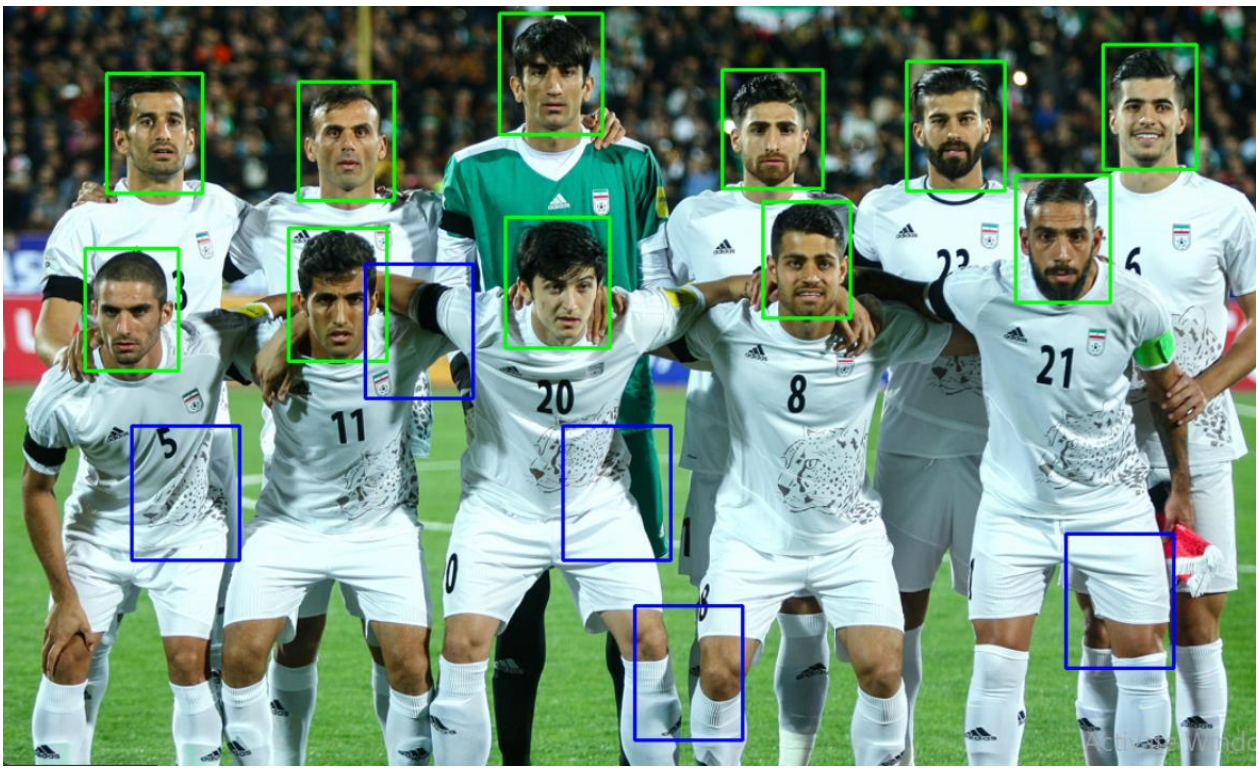
```
window = (90, 112)
step_size = 15
face_proposals = []
back_prpposals = []
for box in sliding_window(img, step_size, window):
    has_face = False
    for face_box in faces_boxes:
        iou = IoU(box, face_box)
        temp_backs = []
        if iou >= 0.7:
          face_proposals.append({'box' : box, 'face' : face_box,
'IoU' : iou})
          has_face = True
          break
        elif iou <= 0.1:
          temp_backs.append(box)
    if not has_face and len(temp_backs) != 0:
        back_prpposals.append(temp_backs[0])
```

به این صورت عمل می‌کنیم که برای هر پنجره‌ای که برای sliding window به دست می‌آوریم مقدار iou را با تمام box های مربوط به صورت محاسبه می‌کنیم. اگر مقدار iou بیشتر از 0.7 باشد این box به طور قطعی یک proposal از کلاس face خواهد بود. اگر iou کمتر از 0.1 باشد به طور موقت به عنوان proposal نمونه از background انتخاب می‌شود و اگر در مراحل بعد به عنوان face proposal انتخاب نشد به عنوان back انتخاب می‌شود.

Proposal های face به شکل زیر هستند:



و ۵ نمونه از back proposal ها نیز به شکل زیر هستند:



۳- سوال سوم

۴- سوال چهارم

Questions

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

2- explain the affection of activation function.

3- explain the affection of using kernel_initializer in layers.

۱) اگر padding را برابر same قرار دهیم در بالا،پایین، چپ و راست تصویرمان به طور یکسان صفر اضافه خواهد شد. حالا اگر در این حالت مقدار stride هم برابر یک باشد، ابعاد تصویر خروجی برابر با ابعاد تصویر ورودی خواهد

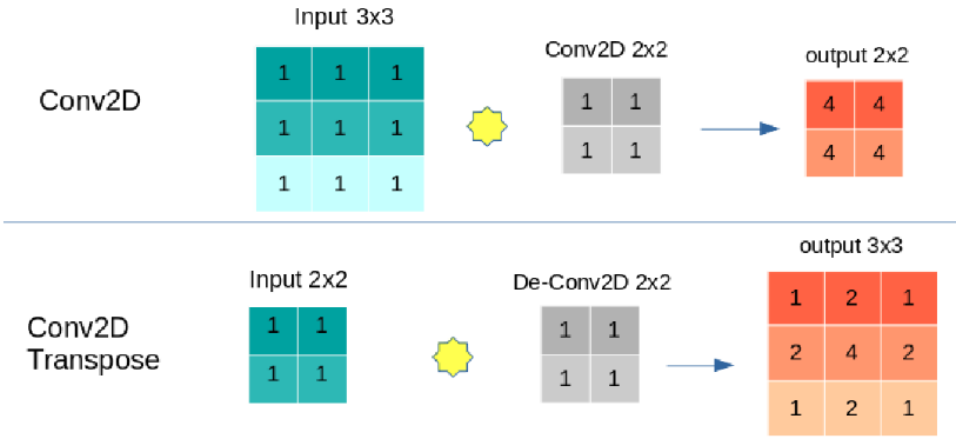
بود. مقدار دیگر برای padding برابر با valid است. در این حالت هیچ padding ای نخواهیم داشت و ابعاد تصویر تغییر خواهد یافت.

۲) تاثیر تابع فعال‌سازی در اینجا درست مانند تاثیر آن در دیگر شبکه‌های عصبی است و خاصیت غیرخطی بودن به شبکه اضافه می‌کند. بدون استفاده از تابع فعال‌سازی مناسب خروجی‌ها همان شکل خطی ورودی‌ها خواهند بود که مفید نیستند.

۳) kernel_initializer وزن‌های فیلترها را با مقدارهای اولیه مناسب می‌سازد. این وزن‌ها می‌توانند تاثیر مهمی در روند آموزش ما داشته باشند و آن را بهتر و سریع‌تر کنند در این سوال از he initializer و توزیع نرمال استفاده شده‌است. می‌توانیم از initializer های دیگر مانند glorot و هم توزیع uniform و هم normal استفاده کنیم..

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

۴) از conv2d عموما برای شناسایی ویژگی‌ها استفاده می‌شود و ورودی را کوچک‌تر می‌کند ولی conv2dtranspose عمل deconv کردن را انجام می‌دهد که بیشتر برای ساختن ویژگی‌ها به کار می‌رود و ورودی را بزرگ‌تر می‌کند. تصویر زیر می‌تواند به خوبی تفاوت این دو عملگر را نشان دهد:



5- explain downsample_block, double_conv_block and upsample_block functions.

۵) ابتدا در مورد double_conv_block صحبت می‌کنیم:

```
def double_conv_block(x, n_filters):
   x = layers.Conv2D(n_filters, 3, padding = "same", activation = "relu",
kernel_initializer = "he_normal")(x)
   x = layers.Conv2D(n_filters, 3, padding = "same", activation = "relu",
kernel_initializer = "he_normal")(x)
   return x
```

این تابع در واقع دو لایه کانولوشنی را پشت سر هم روی ورودی اعمال می‌کند. ورودی ما تصویر یا ماتریس ورودی و تعداد فیلترهایی است که می‌خواهیم روی آن اعمال شود. اندازه کرنل ما ۳*۳ خواهد بود و چون padding برابر same است و stride هم برابر یک است، ابعاد ورودی و خروجی برابر خواهد بود فقط تعداد کانال‌ها برابر تعداد فیلترها خواهد شد و لایه دوم نیز همین کار را تکرار خواهد کرد. پس خروجی همان ابعاد ورودی را خواهد داشت و فقط تعداد کانال‌های آن تغییر خواهد کرد.

Downsample_block به شکل زیر است:

```
def downsample_block(x, n_filters):
    f = double_conv_block(x, n_filters)
    p = layers.MaxPool2D(2)(f)
    p = layers.Dropout(0.3)(p)
    return f, p
```

در واقع در این تابع قصد داریم تصویر ورودی را down scale کنیم. ابتدا double_conv_block روی تصویر اعمال می‌شود که ابعاد آن را حفظ می‌کند ولی تعداد کانال‌های آن را تغییر می‌دهد. در قسمت یک لایه max pooling داریم که کرنل آن ۲*۲ و هم چنین stride آن برابر ۲ خواهد بود که در این صورت ابعاد تصویر ما به جز تعداد کانال‌های آن نصف خواهند شد. سپس یک لایه dropout داریم تا از overfit جلوگیری کنیم. این لایه بعضی از ورودی‌ها را با توجه به احتمالی که به آن دادیم صفر خواهد کرد.

Upsample_block به صورت زیر است:

```
def upsample_block(x, conv_features, n_filters):
    x = layers.Conv2DTranspose(n_filters, 3, 2, padding="same")(x)
    x = layers.concatenate([x, conv_features])
    x = layers.Dropout(0.3)(x)
    x = double_conv_block(x, n_filters)

    return x
```

برخلاف بلاک قبلی این تابع قصد دارد تصویر را up scale کند. در قسمت اول یک لایه conv2dtranspose داریم که ابعاد تصویر را دو برابر خواهد کرد چون مقدار stride برابر ۲ است و padding هم same است. در لایه دوم ویژگی‌هایی که از لایه قبل به دست آوریم را با ویژگی‌های کانولوشنی که به عنوان ورودی داشتیم concat می‌کنیم و روی آن dropout اعمال می‌کنیم. در نهایت هم دوباره conv block اولیه را روی نتایج اعمال می‌کنیم.

6- why use an optimizer in learning?

7- why use compile function?

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

۶) optimizer در بعضی از پارامترهای شبکه مثل نرخ یادگیری یا وزن‌ها تغییر ایجاد می‌کند که این کار باعث می‌شود میزان loss شبکه کاهش یابد و مقدار accuracy زیاد شود که به طور به آموزش بهتر شبکه کمک می‌کند.

۷) برای اینکه یک مدل نهایی شود و بتوانیم از آن برای آموزش استفاده کنیم نیاز به استفاده از تابع compile داریم. در اینجا ما تابع optimizer، تابع loss و metric هایی که می‌خواهیم مدلمان بر اساس آن ارزیابی شود را مشخص می‌کنیم.

۸) زمانی از این تابع loss استفاده می‌کنیم که label های ما ۲ یا بیشتر کلاس داشته باشند. هم چنین label ها باید به صورت one hot باشند.

9 - explain earlystopping function

10 - explain different between fit and compile functions in Keras

11- explain the difference between batch and epoch

۹) تابع earlystopping به عنوان یک شرط پایانی برای آموزش مدل استفاده می‌شود. در حالت عادی مدل تا مقدار epoch ای که مشخص می‌کنیم به آموزش ادامه خواهد داد اما در اینجا ما یک سری metric مشخص می‌کنیم تا در صورتی که این metric دیگر بهبود نیافت یا مقدار بهبود کمتر از مقدار خاصی بود فرآیند آموزش متوقف شود.

۱۰) همان طور که در قبل گفته شد compile فقط موارد اساسی برای آموزش مدل را مشخص می‌کند ولی خودش فرآیند آموزش را انجام نمی‌دهد. در fit ما داده‌های آموزشی را به مدل می‌دهیم و در اینجا آموزش مدل روی داده‌ها شروع خواهد شد و مدل قدم به قدم بهتر کردن وزن‌ها را آغاز می‌کند.

۱۱) در هر epoch ما به طور کامل روی همه داده‌ها train کردن را انجام می‌دهیم ولی هر batch در واقع قسمتی از داده train است که آموزش روی آن انجام می‌شود.