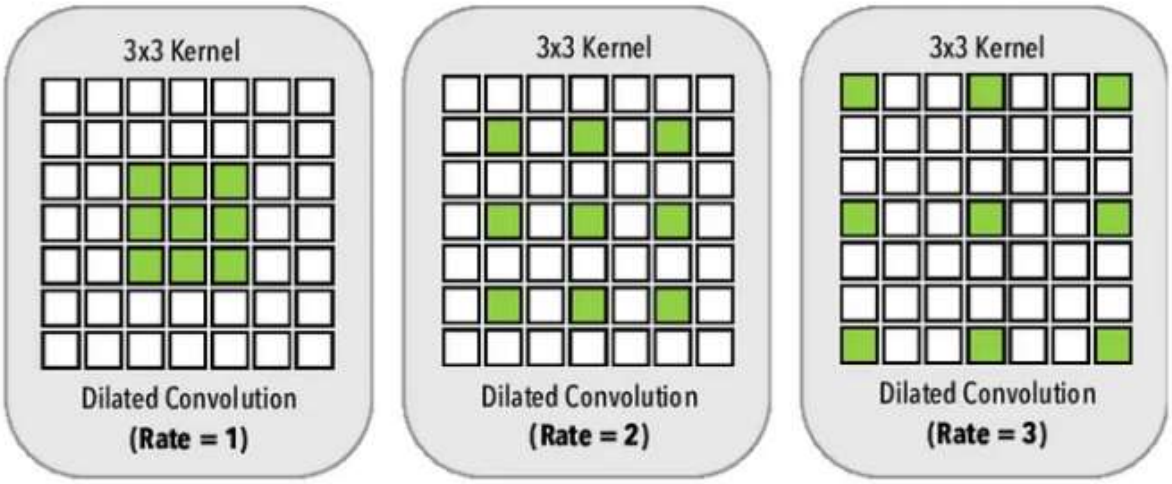
به نام خدا

امیرمحمد کمیجانی ۹۹۵۲۲۰۳۲

سوالات تئوری + توضیحات کد تمرین سری ۷ ام درس بینایی کامپیوتر

۱)

الف) برای محاسبه اندازه dialated kernel از تصویر زیر میتوانیم استفاده کنیم.



میبینیم به ازای اندازه کرنل یکسان 3*3 و با مقادیر مختلف r اندازه پنجره dialated بیشتر شده است.

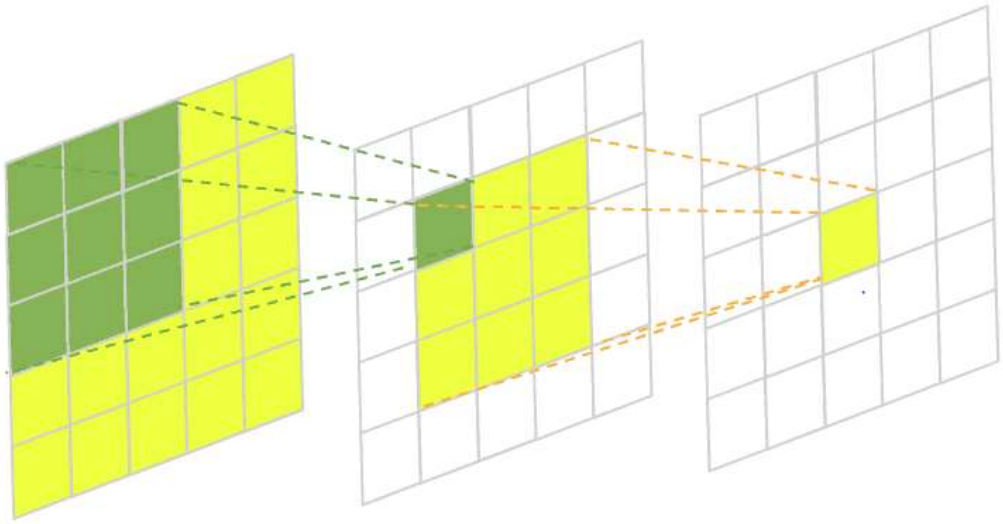
فرمولی که برای کرنل های جدید داریم میتواند به این صورت باشد :

[(k-1) * (d-1) + k] * [(k-1) * (d-1) + k]

حال اگر مقادیر را جایگذاری کنیم مشخص است.برای کرنل ۳*۳ که ثابت است و مقدار k =3 میباشد اگر مقدار d=1 باشد به ما همان 3*3 را میدهد و اگر d=2 باشد به ما یک کرنل ۵*۵ میدهد و ...

ب) تعداد پارامتر های قابل یادگیری تغییری نمیکنند زیرا dialation rate فقط فاصله نقاطی که در کرنل برای محاسبه کانولوشن هستند را بیشتر میکند نه اینکه تعداد نقاط را بیشتر کند.همانطور که در شکل بالا نیز مشخص است تعداد نقاط سبز رنگ کماکان ۹ نقطه میماند و اضافه شدن تعداد پارامترهای قابل یادگیری بستگی به اندازه کرنل یا k دارد نه r.

ج) برای مفهوم receptive field از تصویر زیر استفاده میکنیم:



حال به بررسی جدول داده شده میپردازیم.

در ستون اول، همانطور که طبق تصویر هم میتوانیم ببینیم ابتدا یک ورودی ۱*۱ داشتیم و بعد از کانولوشن به یک ۳*۳ و البته dilation rate =1 تبدیل به یک ۳*۳ نیز شده است.

برای ستون بعدی نیز طبق همین شکل میتوانیم ببینیم که به یک تصویر ۵*۵ تبدیل شده است.

حال برای اینکه یک فرمول کلی داشته باشیم و با مقادیر جدید dilation نیز بتوانیم حساب کنیم از فرمول زیر استفاده میکنیم :

$$s_{l_0} = 1$$

$$s_{l_i} = s_{l_i} + (kernelsize - 1) \cdot dilationfactor$$

در ستون سوم ، dilation = 4 و اندازه کرنل ۳*۳ میباشد پس طبق فرمول مقدار receptive field برابر ۱۳*۱۳ میشود. => A = 13*13

در ستون چهارم ، باید مقدار dilation را بدست آوریم که به صورت زیر میشود:

35 = 13 + (3 - 1) * d => d = 11

در ستون پنجم ، نیز از فرمول میرویم و مقدار c = 51*51 میشود.

برای مابقی ستون ها و مقادیر نیز از فرمول استفاده میکنیم که طریقه استفاده آن در قسمت های قبل بیان شد.

در ستون ششم ، D = 63*63 میشود.

در ستون هفتم ، مقداری برای محاسبه نیست.

در ستون هشتم ، E = 7 میشود.

د) برای این سوال مقدار dilation rate = 1 و مقدار اولیه receptive field را نیز برابر ۱*۱ در نظر گرفتیم. طبق گفته سوال که ۳ لایه pooling و ۲ لایه کانولوشن داریم به این صورت میشود:(طبق پیام تی ای در تلگرام)

.Conv -> conv -> max pool -> max pool -> max pool

مشخصات تمام لایه های max pooling نیز برابر است و کانولوشن را با اندازه کرنل ۵*۵ انجام میدهیم.

حال خواسته مسئله از ما حداقل گام یا حداقل مقدار stride میباشد که با این معماری به receptive field = 107 * 107 برسیم.

طبق فرمولی که در مرحله قبل نیز داشتیم میتوانیم انجام بدهیم و طبق اندازه کرنل که ۵*۵ میباشد بعد از لایه اول receptive field = 5*5 میشود و بعد از لایه دوم برابر با ۹*۹ میشود.حال به لایه های pooling میرویم.
برای مقدار stride در pooling به این صورت است که پنجره ای که طبق آن باید عملیات ماکسیموم گیری را انجام دهیم مشخص میشود و در اصل receptivefield ما به اندازه پنجره stride ضرب میشود یا به اصطلاح n برابر میشود اگر پنجره stride برابر n*n باشد.

از اینجا به بعد سه لایه پولینگ داریم و به این صورت میشود که :

(9 * (n*n*n*)) * (9 * (n*n*n)) = 107 * 107

طبق این موارد حداقل مقدار n = 3 میباشد.

۲)

برای کانولوشن معمولی از فرمول زیر که در جزوه ذکر شده است استفاده میکنیم:

$$- \quad W_2 = (W_1 - F + 2P)/S + 1$$
$$- \quad H_2 = (H_1 - F + 2P)/S + 1$$
$$- \quad D_2 = K$$

برای w2,h2 طبق فرمول و با توجه به اینکه پدینگ نداریم و مقدار stride = 1 میباشد و همچنین مقدار فیلتر جدید یا k=64 است؛ حساب میکنیم:

128 – 5 + 1 = 124

پس در لایه بعدی خواهیم داشت : 128,128,64

برای پارامترهای لایه کانولوشنی:

وز $F \cdot F \cdot D_1 \cdot K$ ند از

داریم 64*3*5*5

برای تعداد ضرب ها نیز داریم :128 * 128 * 5 * 5 * 64 * 3

برای depthwise seperable convolution دو قسمت داریم که باید انجام دهیم ابتدا باید کانولوشن depthwise و سپس کانولوشن pointwise انجام دهیم.

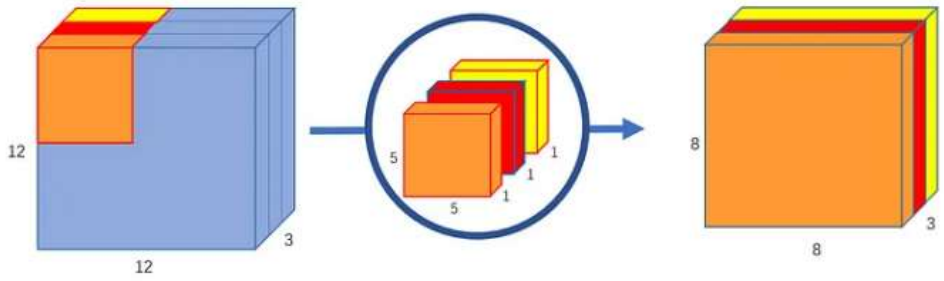


Image 6: Depthwise convolution, uses 3 kernels to transform a 12×12×3 image to a 8×8×3 image

The pointwise convolution is so named because it uses a 1x1 kernel, or a kernel that iterates through every single point. This kernel has a depth of however many channels the input image has; in our case, 3. Therefore, we iterate a 1x1x3 kernel through our 8x8x3 image, to get a 8x8x1 image.

برای توضیح هر کدام مطابق لینک داده شده و این قسمت ها محاسبات را انجام میدهیم.

برای بخش اول ، ابعاد ورودی را در تعداد عمق ورودی ضرب میکنیم، و به صورت 5*5*3 تعداد پارامترهایمان میشود.

برای بخش دوم ، ابعاد به صورت 1*1*3 میشود در نتیجه تعداد پارامتر های ما برابر filter * depth میشود و برابر 64*3 میشود.

حال برای محاسبه تعداد پارامتر های کل این دو مقدار را نیز با هم جمع میکنیم.

برای محاسبه تعداد ضرب ها داریم :

Depthwise = 3 * 5 * 5 * 128 * 128

Pointwise = 3 * 64 * 128 * 128

سپس این دومقدار را نیز با هم جمع میکنیم تا مجموع کل به دست آید.

حال با توجه به مقایسه تعداد پارامتر ها و تعداد ضرب ها میفهمیم depthwise seperable تعداد پارامترها و ضربهای کمتری دارد.

ب)

برای depthwise seperable باز هر بخش را جداگانه محاسبه میکنیم:

Depth wise = 3*3*32

Pointwise = 32 * 32

که جمع این مقادیر برابر

برای کانولوشن معمولی از فرمولی که قبل تر تصویر آنرا گذاشتیم استفاده میکنیم:

3*3*32*32

حال این مقادیر را بر هم تقسیم میکنیم:

1312/9216

که تقریبا در حالت جدید 1/9 شده است.

۳)

طبق جزوه دو متد استفاده شده به این صورت میباشند :

- TM_CCORR $$R(x,y) = \sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))$$

TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$

$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

همانطور که در تصویر میبینیم گاها دو پیکسل کنار هم دارای مقادیر بسیار متفاوت میباشند مثلا ۲۰۰ و ۷. در فرمول دوم که نرمالیزیشن صورت گرفته است ما میتوانیم تصویر را متعادل تر کرده و نویزی بودن تصویر را کاهش دهیم این کار از طریق تقسیم بر انرژی صورت میگیرد(مخرج).

در متد دوم T' استفاده میکنیم که طبق فرمول بالا مشخص است. این مقدار برای این است که ما به شدت نور کمتر حساس بشویم و بیشتر به تغییرات و پترن توجه میکنیم.

در متد اول یا کورلیشن ما بیشتر شباهت را میسنجیم و هر چه بزرگتر بشوند یعنی شباهت بیشتری دارند.

در نتیجه با توجه به موارد بیان شده از متد دوم استفاده میکنیم.

قسمت عملی

ب) برای انجام عملیات تطبیق کلیشه از تابع زیر که در جزوه نیز موجود است استفاده میکنیم :

```
result = cv2. matchTemplate(image, templ, method[, mask])

// image:      Image where the search is running. It must be 8-bit or 32-bit floating-point (W×H)
// templ:      Searched template. It must be not greater than the source image and have the same data type (w×h)
// method:     Parameter specifying the comparison method (cv2.TM_SQDIFF, ..., cv2.TM_CCOEFF_NORMED)
// mask:       Optional mask. It must have the same size as templ. If the data type is CV_8U, the mask is interpreted
               as a binary mask, for data type CV_32F, the mask values are used as weights
// result:     Map of comparison results. It must be single-channel 32-bit floating-point ((W−w+1)×(H−h+1))
```

همانطور که مشخص است به تصویر اصلی و البته یک تصویر به عنوان کلیشه نیاز داریم.تصویر اصلی که همان تصویر سکه ها است که داریم.حال برای کلیشه که تصویر خاصی نداریم باید الگوهایی از تصویر اصلی را به دست آوریم.به دنبال سکه ها هستیم که شکل دایروی دارند و برای اینکه بتوانیم سکه ها را تشخیص دهیم باید از الگوریتم های یافتن اشکال دایروی استفاده کنیم که در فصول قبل خوانده ایم.با استفاده از لبه یاب canny و الگوریتم hough برای اشکال دایروی از تابع houghcircles استفاده میکنیم.سپس بزرگترین دایره را مشخص کرده و تمپلیت خود یا همان کلیشه را طبق آن معین میکنیم.سپس از تابع matchTemplate استفاده میکنیم و اشکالی که از یک مقدار آستانه حد یا threshold بیشتر بودند را معرفی میکنیم.هر چقدر مقدار آستانه را بیشتر کنیم سکه های کمتری قبول میشوند و بالعکس.من در اینجا این مقدار را 0.5 گرفتم و در تصویر خروجی که میبینیم برای همه سکه ها یک فضای مستطیلی قرار داده شده است.

۴)نیاز به داک ندارد.

۵)

برای این سوال سعی میکنم سِل به سِل کد های مهم را بررسی کنم.

```
train_ds = load_voc(split="sbd_train")
eval_ds = load_voc(split="sbd_eval")
```

ابتدا دیتاست مربوط به دیتای آموزش و ارزیابی را لود میکنیم.voc مخفف visually object classes میباشد و برای تسک های ناحیه بندی یا یافتن آبجکت ها میباشد.

```python
def preprocess_tfds_inputs(inputs):
    def unpackage_tfds_inputs(tfds_inputs):
        return {
            "images": tfds_inputs["image"],
            "segmentation_masks": tfds_inputs["class_segmentation"],
        }

    outputs = inputs.map(unpackage_tfds_inputs)
    outputs = outputs.map(keras_cv.layers.Resizing(height=224, width=224))
    outputs = outputs.batch(32, drop_remainder=True)
    return outputs


train_ds = preprocess_tfds_inputs(train_ds)
batch = train_ds.take(1).get_single_element()
keras_cv.visualization.plot_segmentation_mask_gallery(
    batch["images"],
    value_range=(0, 255),
    num_classes=21,
    y_true=batch["segmentation_masks"],
    scale=3,
    rows=2,
    cols=2,
)
```

در تابع درونی یک دیکشنری از تصاویر و مَسک ناحیه آنها که در دیتاست ورودی میباشد ، خروجی میدهیم.

سپس شروع به عملیات preprocessing میکنیم؛عملیات هایی مثل resize کردن و البته batch کردن.

در قسمت بعدی نیز خروجی از کارهایی که انجام دادیم را نمایش میدهیم.

```python
eval_ds = preprocess_tfds_inputs(eval_ds)
```

```python
train_ds = train_ds.map(keras_cv.layers.RandomFlip())
train_ds = train_ds.map(keras_cv.layers.RandomRotation(factor=.1,segmentation_classes=21))

batch = train_ds.take(1).get_single_element()

keras_cv.visualization.plot_segmentation_mask_gallery(
    batch["images"],
    value_range=(0, 255),
    num_classes=21,
    y_true=batch["segmentation_masks"],
    scale=3,
    rows=2,
    cols=2,
)
```

عملیات preprocess را روی دیتای ارزیابی نیز انجام میدهیم.
سپس با استفاده از randomFlip,randomRotation به نوعی عملیات data augmentation را انجام میدهیم.
سپس عملیات visualization را همانند سِل قبل بر روی دیتای ترین جدید انجام میدهیم.

در قسمتی بعدی که باید کامل میکردیم باید طبق معماری شبکه unet مدل خود را میساختیم.برای این معماری و انجام لایه های کانولوشنی از تصویر زیر استفاده کردیم.

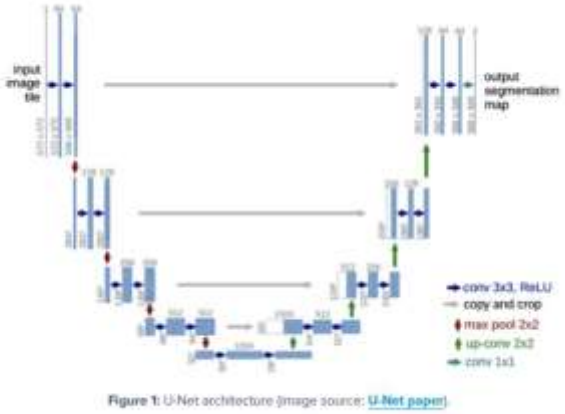
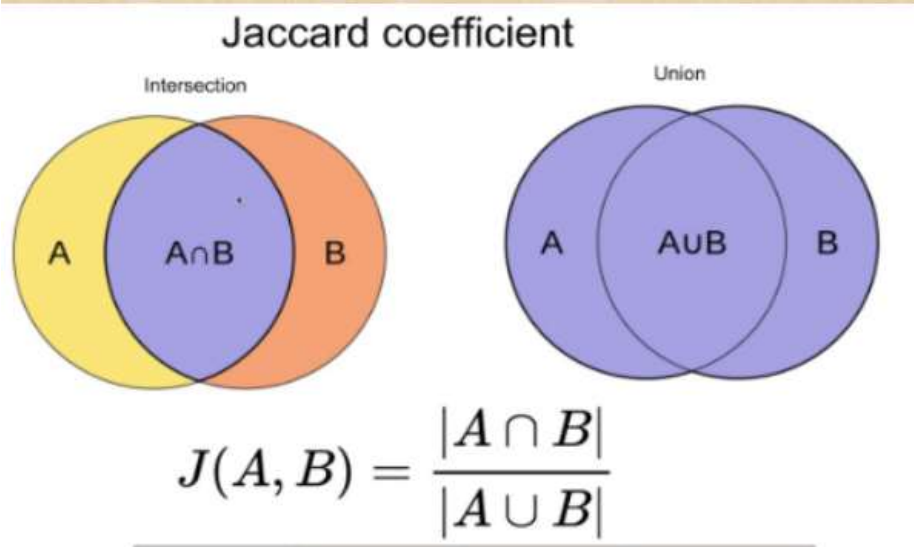


Figure 1: U-Net architecture (image source: U-Net paper).

این فایل را ضمیمه کرده ام.

```
dice_loss = sm.losses.DiceLoss()
focal_loss = sm.losses.CategoricalFocalLoss()
total_loss = dice_loss + (1 * focal_loss)
```

برای محاسبه loss خود از DiceLoss و FocalLoss استفاده کرده ایم که به ترتیب برای loss بین مَسک یک ناحیه طبق پیشبینی ما و ناحیه درست است و در نهایت با ادغام این دو مقدار مجموع loss را بدست می آوریم.



برای قسمت بعدی که jaccard coef هست از این تصویر استفاده کرده ایم.فقط ابتدا دیتا را به صورت flatten تبدیل کردیم و سپس از فرمول استفاده کردیم.همچنین K نیز برای انجام عملیات های مختلف ریاضی بر روی تنسور ها میتواند مفید باشد.

در قسمت بعد معیارهای عملکرد مدل را برابر دقت و jaccard coef قرار دادیم و سپس مدل را با استفاده از بهینه ساز adam و البته تابع ضرری که در قسمت قبل معرفی کردیم انجام و معیار های مشخص کامپایل کردیم و سپس خلاصه ای از مدل را نمایش دادیم و در مرحله بعد آموزش مدل را آغاز کردیم.

در قسمت بعدی که میخواهیم از یک مدل از پیش تعیین شده استفاده کنیم ابتدا متغیرهای خود را معین میکنیم سپس از تابع Unet و backbone ی که مشخص شده به همراه وزن هایی که در imagenet داشته ایم استفاده میکنیم.سپس مدل را کامپایل میکنیم و در مرحله بعد انکودر را فریز میکنیم و اجازه میدهیم تنها دیکودر آموزش ببیند و آموزش را آغاز میکنیم.در آخر نیز اجازه میدهیم کل شبکه آموزش ببیند.این کار را با trainable کردن تمام لایه ها انجام میدهیم.

۶)

ابتدا دو تابع download,extract را نوشتیم برای دریافت دیتاست pascal voc نوشتیم.دقت شود از دیتاست سال ۲۰۰۸ استفاده کردیم به دلیل حجم کمتر به نسبت دیتاست سال ۲۰۱۲. در تابع بعدی که برای فایل های xml موجود در دایرکتوری Annotations را یافتیم و طبق فایلهای این دایرکتوری مسیر عکس ها و لیست box ها و لیست لیبل ها را به دست می آوریم.در سِل بعدی تعداد اجزای هر یک را به عنوان دیباگ و چک کردن کد انجام داده ایم.

در بخش سوم تابع visualize از قبل آماده و مناسب است و برای پیدا کردن تصاویر و البته باکس بین آبجکت های یافت شده است.فقط در قسمت آخر از یک حلقه for استفاده کردم تا این کار را برروی ۱۰ تصویر انجام دهیم و لیبل هر یک را مشخص کرده ایم.(طبق خواسته داک)

در بخش چهارم از مدلی که در هاب تنسورفلو بود استفاده کردیم که برای faster-rcnn میباشد.

در بخش بعدی باید عملیات preprocessing را انجام دهیم؛ابتدا مسیر یک عکس را در ورودی میدهیم و سپس عکس را میخوانیم.در مرحله بعد عکس را به کانال RGB میبریم و سپس عملیات resize کردن را انجام میدهیم.اندازه عکس ها را به ۶۴۰*۶۴۰ تبدیل کردیم به این دلیل که طبق سرچی که انجام دادم متوجه شدم مدل zoo برروی عکس هایی با این اندازه آموزش دیده شده است.

I am watching the list of all tensorflow2 Zoo Model. Assuming that 640x640 is the size of image, I was wondering what happen if the input image is bigger than the model size.

For example if we use :

```
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)
```

لینک : https://stackoverflow.com/questions/66481008/tensor-flow-zoo-model-object-detection-does-size-affect-result

و سپس برای اینکه شکل عکس مطابق با مدلی که از آن استفاده کردیم شود ابعاد آنرا گسترش دادیم و در نهایت عکس پیش پردازش شده و عکس اصلی را ریترن کردیم.البته در اسم مدل که در قسمت قبل نیز بود هم مشخص است که باید اندازه عکس ها به ۶۴۰*۶۴۰ تبدیل شوند.

در مرحله بعد، برای پیدا کردن آبجکت ها ابتدا پیش پردازش را انجام میدهیم؛سپس عکس پیش پردازش شده را به مدل خود میدهیم.در detections که یک دیکشنری میباشد دارای کلید های متنوعی هستیم که برای این تَسک از سه کلید استفاده میکنیم. مختصات باکس هایی که برای تشخیص آبجکت ها میباشند ، ضریب اطمینان یا همان score که مشخص میکند که ما یک آبجکت را تشخیص بدهیم یا خیر و همچنین آیدی مربوط به کلاس مربوط به هر آبجکت که در انتها هر آبجکت را بر اساس این کلاس مشخص کرده ایم.

The output dictionary contains:

- `num_detections`: a `tf.int` tensor with only one value, the number of detections `[N]`.
- `detection_boxes`: a `tf.float32` tensor of shape `[N, 4]` containing bounding box coordinates in the following order: `[ymin, xmin, ymax, xmax]`.
- `detection_classes`: a `tf.int` tensor of shape `[N]` containing detection class index from the label file.
- `detection_scores`: a `tf.float32` tensor of shape `[N]` containing detection scores.
- `raw_detection_boxes`: a `tf.float32` tensor of shape `[1, M, 4]` containing decoded detection boxes without Non-Max suppression. `M` is the number of raw detections.
- `raw_detection_scores`: a `tf.float32` tensor of shape `[1, M, 90]` and contains class score logits for raw detection boxes. `M` is the number of raw detections.
- `detection_anchor_indices`: a `tf.float32` tensor of shape `[N]` and contains the anchor indices of the detections after NMS.
- `detection_multiclass_scores`: a `tf.float32` tensor of shape `[1, N, 90]` and contains class score distribution (including background) for detection boxes in the image including background class.

در قسمت بعدی مختصات باکس ها را بر اساس x,y کمینه و بیشینه محاسبه میکنیم؛و در ارتفاع و عرض تصویر اصلی ضرب میکنیم تا این مختصات را با scale تصویر اصلی در باکس ها قرار دهیم و در انتها یک مستطیل به دور آبجکت ها با استفاده opencv رسم میکنیم و همچنین برای لیبل گذاری را نیز بر اساس کلاس هایی که داشتیم انجام دادیم.

مدلی که از آن استفاده کردیم طبق داکیومنت موجود در تنسورفلو،از دیتاست coco استفاده شده است:

https://www.kaggle.com/models/tensorflow/faster-rcnn-resnet-v1/tensorFlow2/faster-rcnn-resnet50-v1-640x640/1?tfhub-redirect=true

همچنین برای متریک های خواسته شده به جز mAP از کتابخانه sklearn.metrics استفاده کرده ام.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

برای Evaluate کردن مدل ابتدا کارهایی که در قسمت شناسایی آبجکت داشتیم را انجام دادم سپس در مرحله بعد باکس های درست که در دیتای validation داریم را با پیش بینی خود مقایسه میکنیم.برای mAP نیز با استفاده از منحنی precsion

recall از مقدار average precisions را محاسبه کردیم سپس در مرحله بعد از آنها میانگین گرفتیم تا مقدار mAP محاسبه شود.

برای تفسیر نتایج نیز با توجه به اینکه از دیتاست coco استفاده میکنیم؛از داکیومنت این سایت استفاده کرده ایم :

فرمت دیتا در coco برای تسک object detection به این صورت میباشد:

```
annotation{
   "id"                : int,
   "image_id"          : int,
   "category_id"       : int,
   "segmentation"      : RLE or [polygon],
   "area"              : float,
   "bbox"              : [x,y,width,height],
   "iscrowd"           : 0 or 1,
}

categories[{
   "id"                : int,
   "name"              : str,
   "supercategory"     : str,
}]
```

همچنین برای فرمت جوابی که ما داریم نیز به این صورت میشود :

For detection with bounding boxes, please use the following format:

```
[{
   "image_id"          : int,
   "category_id"       : int,
   "bbox"              : [x,y,width,height],
   "score"             : float,
}]
```

Note: box coordinates are floats measured from the top left image corner (and are 0-indexed). We recommend rounding coordinates to the nearest tenth of a pixel to reduce resulting JSON file size.

که به این فرمت میباشد و یک آیدی عکس که با توجه به عکس های داخل دیتاست میباشد سپس یک آیدی مناسب دسته بندی یا category برای بحث لیبل گذاری آبجکت های شناسایی شده و باکس هایی که آبجکت ها را درون آنها قرار میدهیم

شامل مختصات نقاط و همچنین یک امتیاز یا Score که اگر از یک آستانه ای بزرگتر بود آن آبجکت را تشخیص میدهیم.