تمرین سری نهم درس تصویربرداری رقمی

پوریا محمدی نسب

(۴۰۰۷۲۲۱۳۸)

۱- شبکه های زیر را با توجه به اجزا تشکیل دهنده و نحوه کار کرد آنها آن از نظر سرعت و دقت و چند مقیاسه(multi scale) بودن نسبت به هم بررسی کنید.

- RCNN
- Fast-RCNN
- Faster RCNN
- Yolo
- SSD
- RetinaNet

در R-CNN از ۲۰۰۰ پروپوزال در تصویر به جای جستجو در تمام تصویر استفاده میشود. این ۲۰۰۰ پروپوزال با الگوریتمی تحت عنوان Selective search استخراج میشوند. حال این پروپوزال ها با یک CNN فیچرها را استخراج میکند که خرجی یک وکتور با عمق ۴۰۹۶ است که این وکتور به یک ماشین SVM پاس داده میشود تا object های درون پروپوزال ها را classify کند. همچنین برای مرز بندی دقیق تر object در پروپوزال الگوریتم ۴ مقدار x,y,w,h را پیشبینی میکند. مشکلات R-CNN همچنان زمان زیاد training است زیرا برای هر تصویر باید 2000 پروپوزال کلاس بندی شود. همینطور برای کاربرد real time مناسب نیست زیرا برای هر تصویر test حدود ۴۷ ثانیه زمان میبرد. و در انتها قسمت selective search یک الگوریتم fix شده است پس در این مرحله یادگیری صورت نمیگیرد و این میتواند باعث پیشنهاد های بد و بی منطق در هر مرحله از پروپوزال ها باشد.

Fast R-CNN در واقع نسخه سریع شده الگوریتم قبلی است و تنها تفاوت عمده ی این معماری این است که به جای پاس دادن پروپوزال ها به لایه Convolution، خود تصویر به این لایه پاس داده میشود تا feature map های کانولوشنی را تولید کند. از این مرحله ما میتوانیم ناحیه ی پروپوزال ها را تشخیص دهیم و همه ی آنها را به یک سایز برابر تبدیل کینم و به یک لایه FC وارد کنیم. دلیل سرعت بیشتر Fast R-CNN نسبت به R-CNN این است که در این الگوریتم مجبور به استخراج ۲۰۰۰ پروپوزال و وارد کردن آن به یک لایه ی Convolution برای هر تصویر نیستیم. اما در کنار سرعت بالای این الگوریتم، پروپوزال ناحیه ها قسمت ضعف این الگوریتم هستند و باعث میشوند که این الگوریتم عملکرد ضعیفی داشته باشد.

دو الگوریتم R-CNN و Fast R-CNN برای پیدا کردن نواحی پروپوزال از الگوریتم selective search استفاده میکنند که الگوریتم زمانبری است پس طراحان الگوریتم Faster R-CNN تصمیم گرفتند که این الگوریتم را از پروسه تشخیص object حذف کنند و به شبکه اجازه دهند که به طور خودکار نواحی پروپوزال را بیابد. در این الگوریتم مانند R-CNN تصویر به یک شبکه کانولوشنی داده میشود تا feature map ها استخراج شوند اما پس از استخراج feature map ها آنها را به selective search پاس نمیدهد و به جای اینکار یک شبکه کانولوشنی جداگانه طراحی کردند تا نواحی پروپوزال را پیشبینی کند و پس از reshape کردن خروجی ها آنها را به یک الگوریتم classification پاس میدهند. با توجع به سرعت فوق العاده این الگوریتم، میتوان از آن در کاربردهای real time نیز استفاده کرد.

الگوریتم هایی که تا به حال بررسی کردیم به صورت محلی فقط به قسمت هایی از تصویر توجه میکردند و کل تصویر را در نظر نمیگرفتند. الگوریتم YOLO (You Only Look Once) الگوریتمی است که با سایر الگوریتم های region-based بسیار متفاوت است. در ای الگوریتم یک شبکه کانولوشنی محدوده اشیاء را پیشبینی میکند. در واقع الگوریتم YOLO تصویر را به یک شبکه S*S تبدیل میکند که در هر خانه از این شبکه تعداد m محدوده وجود دارد. برای هر محدوده الگوریتم probability و offset تعیین میکند و جعبه هایی که شانسی بیشتر از حد آیتانه مشخص شده داشته باشند به عنوان object تشخیص داده میشوند. الگوریتم YOLO چندین برابر از سایر الگوریتم ها سریع تر است ولی ضعف آن در کارکردن با اشیاء کوچک داخل تصویر است برای مثال برای تشخیص دسته ای از پرندگان در یک تصویر مشکل دارد.

RetinaNet یک مدل تشخیص اشیاء تک مرحله ای است که از loss ای به نام focal loss برای بررسی عدم تعادل در کلاس ها در حین training استفاده میکند. بعلاوه در این مدل یک بهبود دیگر نیز بعمل آمده است که استفاده از Feature Pyramid Networks(FPN) است. معماری retinanet از یک backbone شامل ResNet و FPN تشکیل شده است و دو لایه نیز یکی برای classification و یکی برای regression استفاده شده است.

و آخرین مدلی که بررسی میکنیم مدل SSD نام دارد.در این الگوریتم نیز تصویر به grid هایی تقسیم میشود با این تفاوت که هر grid مسئول تشخیص object داخل خودش است. اگر شی ای در یک خانه از شبکه نباشد آن را به عنوان background در نظر میگیریم. حال اگر در یک ناحیه تعداد بیشتری شی وجود داشته باشد از تکنیک Ancher Box استفاده میکنند.

۲- در الگوریتمهای تشخیص اشیاء دو مرحلهای، از یک بخش برای استخراج جعبههای پیشنهادی استفاده میشود. این بخش در برخی الگوریتمها مانند Faster RCNN آموزش میبیند و در برخی الگوریتمها مانند Fast RCNN از یک الگوریتم ثابت مانند selective search ، bing، edge boxes و... استفاده میکنند؛ با خواندن مقاله زیر ایده edge boxes را توضیح دهید.

Zitnick, C. Lawrence, and Piotr Dollár. "Edge boxes: Locating object proposals from edges." European conference on computer vision. Springer, Cham, 2014.

در این مقاله برای قسمت تولید bounding box پروپوزال ها از متدی تحت عنوان edge استفاده شده است. این تکنیک از شاخص اندازه گیری objectness score استفاده میکند که عبارت است از تعداد تعداد لبه های یک box منهای آن تعدادی که با مرز box همپوشانی دارند. در این الگوریتم در ابتدا پروپوزال های اشیا بر اساس کانتورهای موجود در bounding box مرتب میشود. سپس تابه اندازه گیزی edge based score تعریف میشود که در ادامه ی الگوریتم اساس کار الگوریتم بر پایه این تابع است. و در قدم آخر، با تکنیک پنجره لغران پروپوزال هایی با بالاترین امتیاز بدست آمده، مشخص میشوند و scale،position و ratio آنها تعیین میشود.

۳ - فایل اکسل داده شده حاوی دو sheet برای مختصات جعبه های مطلوب و پیشبینی شده است. با توجه به مقادیر داده شده AP75 ، AP50 و Ap25 را محاسبه کنید. (رسم نمودار precision - recall الزامی است)

برای حل این سوال و انجام بعضی محاسبات از کد استفاده میکنیم.

```
def wh2xy(Box):
    x1 = Box[0]
    y1 = Box[1]
    x2 = Box[0] + Box[2]
    y2 = Box[1] + Box[3]

    return [x1,y1,x2,y2]
```

تابع wh2xy چهارتایی (x1,y1,width,height) را به چهارتایی (x1,y1,x2,y2) تبدیل میکند.

```
def intersection_over_union(boxA, boxB):
    # determine the (x, y)-coordinates of the intersection rectangle

    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])

    # compute the area of intersection rectangle
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
    # compute the area of both the prediction and ground-truth
    # rectangles
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1] + 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1] + 1)
    # compute the intersection over union by taking the intersection
    # area and dividing it by the sum of prediction + ground-truth
    # areas - the interesection area
    iou = interArea / float(boxAArea + boxBArea - interArea)
    # return the intersection over union value
    return iou
```

تابع intersect_over_union به عنوان ورودی دو مربع گرفته و IOU را محاسبه میکند.

```
matrix_of_IOU = []

for i in range(len(ground_truth)):
    tmp = []
    for j in range(len(pred)):
        tmp.append(intersection_over_union(G_T.loc[i].values,pred.loc[j].values))
    matrix_of_IOU.append(tmp)
```

```
matrix_of_IOU
```

```
[[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
 [0.0, 0.0, 0.0, 0.6864406779661016, 0.0, 0.0, 0.0, 0.0, 0.06769230769230769],
 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.859441151566469, 0.0, 0.0],
 [0.0, 0.0, 0.0, 0.0, 0.0, 0.2860696517412935, 0.0, 0.0, 0.0],
 [0.924924924924925, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]]
```

قطعه کد بالا با استفاده از دو تابع تعریف شده ی بالا، مقدار IOU را به ازای همه ی prediction ها برای تک تک ground truth ها محاسبه میکند.

```
img = np.ones((210,200,3),dtype=np.uint8)
img = img * 255
```

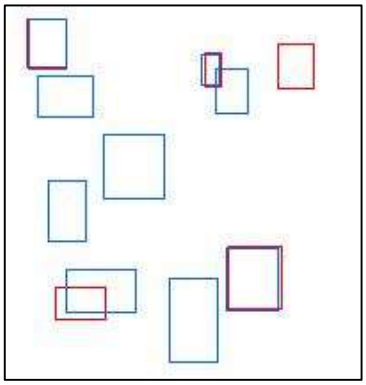
```
for row in G_T.values:
    cv2.rectangle(img,(int(row[0]),int(row[1])),(int(row[2]),int(row[3])),color=(255,0,0))

plt.imshow(img)

for row in pred.values:
    cv2.rectangle(img,(int(row[0]),int(row[1])),(int(row[2]),int(row[3])),color=(0,100,255))

plt.figure()
plt.imshow(img)
```

سپس با نمایش ground truth ها و prediction ها درک بهتری از اعداد بدست آمده به صورت بصری خواهیم داشت.



حال با داشتن مقادیر IoU میتوان True یا False بودن prediction ها به ازای threshold های مختلف را محاسبه کرد.
جدول زیر تمام حالت به ازای threshold های متفاوت را نشان میدهد.

IOU	ground truth		Th = 0.25	Th = 0.5	Th = 0.75
0.92	5		T	T	T
0	-		F	F	F
0	-		F	F	F
0.69	2		T	T	F
0	-		F	F	F
0.29	4		T	F	F
0.86	3		T	T	T
0	-		F	F	F
0.07	2		F	F	F

در انتها میتوان با استفاده از دو فرمول زیر مقادیر precision و recall را محاسبه کرد.

$$Precision = \frac{TP}{\text{total positive results}}$$

$$Recall = \frac{TP}{\text{total cancer cases}}$$

برای محاسبه نهایی این مقادیر ابتدا آنها را بر اساس Score داده شده Sort میکنیم و با threshold های مختلف مقادیر محاسبه میشود.
برای threshold = 0.25 جدول زیر را داریم:

index	score	TH = 0.25	Precision	Recall
4	0.96	T	1	0.2
3	0.89	T	1	0.4
1	0.84	F	0.66666667	0.4
2	0.79	T	0.75	0.6
7	0.74	T	0.8	0.8
9	0.62	F	0.66666667	0.8
5	0.47	T	0.71428571	1
6	0.39	F	0.625	1
8	0.29	F	0.55555556	1

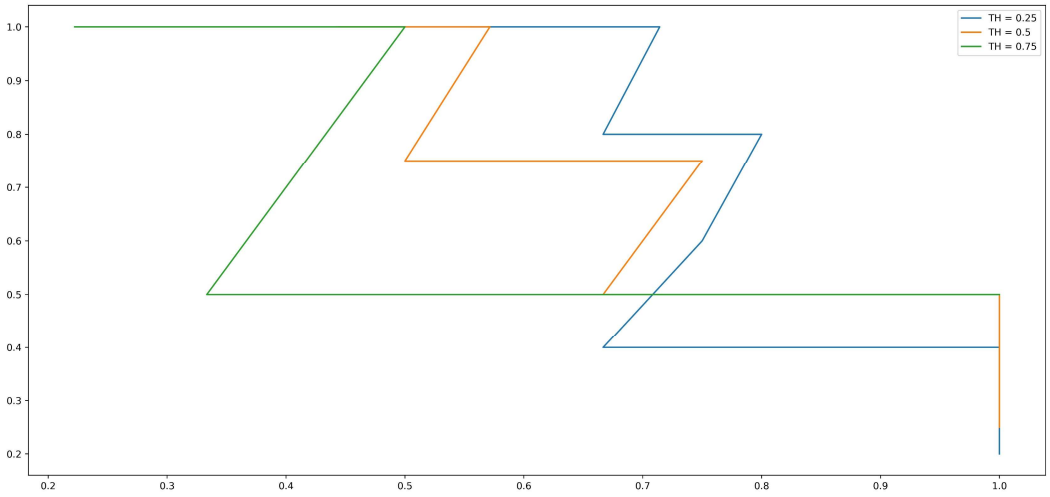
جدول بدست آمده برای threshold = 0.5:

4	0.96	T	1	0.25
3	0.89	T	1	0.5
1	0.84	F	0.66666667	0.5
2	0.79	T	0.75	0.75
7	0.74	F	0.6	0.75
9	0.62	F	0.5	0.75
5	0.47	T	0.57142857	1
6	0.39	F	0.5	1
8	0.29	F	0.44444444	1

جدول بدست آمده برای threshold = 0.75:

4	0.96	T	1	0.5
3	0.89	F	0.5	0.5
1	0.84	F	0.33333333	0.5
2	0.79	T	0.5	1
7	0.74	F	0.4	1
9	0.62	F	0.33333333	1
5	0.47	F	0.28571429	1
6	0.39	F	0.25	1
8	0.29	F	0.22222222	1

حال میتوان نمودار precision-recall را برای هر سه Threshold رسم کرد.



۵- ب) مقاله YOLOv4 را مطالعه کنید و کارهای انجام شده برای افزایش دقت نسخه اولیه YOLO را توضیح دهید.

در مقایسه با ورژن اصلی YOLO، مدل YOLOv4 دارای مزایایی است که در زیر به آنها اشاره میکنیم:

- از دو تکنیک و الگوریتم بسیار جدید با نام های Bag-of-Freebies و Bag-of-Specials در طول زمان train استفاده شد. این در الگوریتم باعث بهبود بسیار زیادی در عملکرد الگوریتم شدند.
- بهینه سازی هایی جهت اجرای بهتر الگوریتم روی GPU ها انجام شد. برای مثال نسخه ۴ الگوریتم برای GPU های 1080 Ti و 2080 Ti نتایج بسیار خوبی نشان داد.
- از متدهای جدیدی مانند CBN(Cross-iteration batch normalization)و PAN(Path aggregation network) برای بهبود عملکرد الگوریتم روی GPU های تک هسته ای استفاده شد.

نویسندگان و طراحان ورژن چهارم YOLO ، موارد زیر را معرفی و در الگوریتم از آنها استفاده کردند:

- Weighted-Residual-Connections (WRC)
- Cross-Stage-Partial-Connections (CSP): که در واقع یک backbone جدی برای مدل است.
- Cross mini-Batch Normalization (CmBN): که این متد در قسمت batch normalization مدل بسیار موثر واقع میشود.
- Self-adversarial-training (SAT): یک تکنیک موثر برای data augmentation است.
- Mish-activation: یک تابع فعال ساز جدید و بسیار کارا در عمل میباشد
- Mosaic data augmentation: این مورد نیز یک متد data augmentation است که ۴ تصویر train را به یک تصویر میکس میکند.
- DropBlock regularization: یک متد regularization جدید است که برای CNN ها بهتر عمل میکند.
- CIoU loss: این loss جدید با سرعت و دقت بالاتری converge میکند.

References

1) towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76
2) towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
3) jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359
4)paperswithcode.com/method/retinanet#:~:text=RetinaNet%20is%20a%20one%2Dstage,learning%20on%20hard%20negative%20examples.
5) https://towardsmachinelearning.org/retinanet-model-for-object-detection-explanation/
6) https://developers.arcgis.com/python/guide/how-ssd-works/
7) syncedreview.com/2020/04/27/yolo-is-back-version-4-boasts-improved-speed-and-accuracy/
8) morioh.com/p/9bca8f92d016