

# به نام خالق رنگین کمان

ستاره باباجانی – 99521109- گزارش تمرین سری هشتم

سوال 1: الف) روش sliding window شامل جابجایی یک پنجره با اندازه ثابت بر روی تصویر ورودی و اعمال یک CNN برای هر منطقه پنجره به طور مستقل است. این روش از نظر مفهومی ساده است. این روش هزینه محاسباتی بالایی دارد زیرا CNN برای هر پنجره ممکن در تصویر اعمال می شود. برای یک تصویر با اندازه

$N \times N$  و یک پنجره با اندازه  $W \times W$ ، تعداد پنجره برای محاسبه  $(N-W+1) \times (N-W+1)$  میباشد. همچنین این روش محاسبات اضافی دارد (Redundancy) زیرا همپوشانی پنجره ها منجر به محاسبات اضافی برای مناطقی از تصویر می شود که توسط چندین پنجره پوشانده شده است. استنتاج آهسته ای نیز دارد و برای کاربردهای real-time مناسب نیست. از مزایای این روش میتوان به سادگی (درک و پیاده سازی آسان) و انعطاف پذیری (می تواند با هر CNN از پیش آموزش دیده بدون تغییر استفاده شود)، اشاره کرد.

از سوی دیگر، بدون استفاده از آن همانند چارچوب های تشخیص شی مدرن مثل SSD، YOLO، و Faster R-CNN از تکنیک هایی مانند شبکه های کاملاً کانولوشنال (FCN) و شبکه های پیشنهادی منطقه (RPNS) برای پیش بینی مستقیم جعبه های مرزی و احتمالات کلاس استفاده میشود. در این حالت کارایی محاسباتی بالا میرود زیرا این روش ها کل تصویر را به یکباره پردازش می کنند و از ماهیت کانولوشنی CNN ها برای به اشتراک گذاشتن محاسبات در کل تصویر استفاده می کنند. همچنین امکان پردازش موازی میدهند که سرعت خیلی زیاد میشود. و همچنین این نوع

طراحی برای کاربرد های real-time مناسب هستند. از معایب این روش میتوان به پیچیدگی (پیاده سازی پیچیده تر و نیاز به تکنیک های آموزشی پیچیده تر) و وابسته بودن به معماری (به شدت به معماری خاص و انتخاب های طراحی وابسته است و انعطاف پذیری آن را نسبت به رویکرد sliding window کمتر می کند)، اشاره کرد.

ب) همان طور که میدانیم YOLO یک چارچوب تشخیص شی است که جعبه های محدود و احتمالات کلاس را مستقیماً از تصاویر کامل در یک ارزیابی واحد پیش بینی می کند.

- مدیریت چندین کلاس در یک سلول شبکه ای: YOLO تصویر ورودی را به یک شبکه  $S \times S$  تقسیم می کند. سپس هر سلول شبکه تعداد ثابتی از جعبه های مرزی (B) را پیش بینی می کند. هر پیش بینی جعبه مرزی شامل مختصات  $(x, y, w, h)$  امتیاز اطمینان و احتمالات کلاس است که احتمالات کلاس یعنی برای هر سلول شبکه، YOLO احتمالات کلاس را برای هر کلاس ممکن پیش بینی می کند. احتمالات کلاس پیش بینی شده مشروط به سلول شبکه حاوی یک شی است.

- مکانیسم پیش بینی: YOLO مرکز  $(x, y)$ ، عرض  $w$  و ارتفاع  $h$  جعبه های مرزی را نسبت به سلول شبکه پیش بینی می کند. سپس از امتیاز اطمینان (نشان دهنده اطمینان مدل است که جعبه مرزی حاوی یک شی و دقت مختصات جعبه مرزی است.) و احتمالات کلاس (احتمالات کلاس مستقل از کادر محدود پیش بینی می شوند و احتمال حضور هر کلاس در سلول شبکه را نشان می دهند)، برای پیش بینی استفاده استفاده میکند.

- جعبه های لنگر: جعبه های لنگر جعبه های مرزبندی از پیش تعریف شده با اشکال و اندازه های مختلف هستند که به عنوان مرجع برای پیش بینی جعبه های مرزی مورد استفاده قرار می گیرند. با استفاده از جعبه های لنگر متعدد با نسبت ها و اندازه های مختلف، YOLO می تواند اشیاء با اشکال و اندازه های مختلف را به طور موثرتری اداره کند. هر سلول شبکه می تواند با تنظیم جعبه های لنگر میتواند چند bounding boxes را پیش بینی کند. این به مدل اجازه می دهد تا در تشخیص اشیایی که با شکل و اندازه جعبه های لنگر مطابقت دارند، تخصص پیدا کند. جعبه های لنگر با ارائه نقطه شروع دقیق تری برای پیش بینی جعبه های مرزی به محلی سازی بهتر اشیاء کمک می کنند.

مزایای استفاده از انکر باکس:

1. عملکرد بهتر: دقت تشخیص بهبود یافته برای اشیاء با اندازه ها و نسبت های مختلف.
2. کارایی: همگرایی سریع تر در طول آموزش و تعمیم بهتر به داده های دیده نشده.
3. مقیاس پذیری: توانایی مدیریت طیف گسترده ای از اندازه های شی در یک چارچوب.

سوال 2: الف) مقایسه ها به شرح زیر هستند:

1. معماری:

- YOLO: YOLO تصویر ورودی را به یک شبکه  $S \times S$  تقسیم می کند. هر سلول شبکه bounding boxes، امتیازات اطمینان آن جعبه ها و احتمالات کلاس را پیش بینی می کند. YOLO bounding boxes و احتمالات کلاس را

مستقیماً از تصاویر کامل در یک ارزیابی واحد پیش‌بینی می‌کند، و آن را به یک مدل واحد برای طبقه‌بندی و محلی‌سازی تبدیل می‌کند.

YOLO یک آشکارساز تک مرحله‌ای است، به این معنی که مستقیماً احتمالات کلاس و مختصات جعبه مرزی را از تصاویر کامل در یک عبور از شبکه پیش‌بینی می‌کند.

نسخه‌های قبلی YOLO از معماری سفارشی مبتنی بر GoogLeNet استفاده می‌کردند. نسخه‌های بعدی، مانند YOLOv3، از Darknet-53 استفاده می‌کنند که معماری عمیق‌تری با الهام از ResNet است.

- SSD: SSD از یک شبکه پایه (به عنوان مثال، VGG16) استفاده می‌کند و به دنبال آن لایه‌های کانولوشنی اضافی که به تدریج اندازه آنها کاهش می‌یابد، هستند. SSD پیش‌بینی‌هایی را در مقیاس‌های مختلف از لایه‌های مختلف شبکه انجام می‌دهد. این به SSD اجازه می‌دهد تا اشیاء با اندازه‌های مختلف را به طور موثرتری مدیریت کند. مشابه جعبه‌های لنگر در YOLO، SSD از جعبه‌های پیش‌فرض (پیش‌های) نسبت‌ها و مقیاس‌های مختلف در هر مکان نقشه‌ویژگی استفاده می‌کند. مانند YOLO، SSD نیز یک آشکارساز تک مرحله‌ای است که به طور مستقیم جعبه‌های مرزی و احتمالات کلاس را پیش‌بینی می‌کند.

2. سرعت:

- یولو: YOLO به داشتن سرعت بالای خود شناخته شده است و می‌تواند تصاویر را در زمان واقعی پردازش کند (به عنوان مثال، YOLOv3 می‌تواند بیش از 30 فریم در ثانیه را در یک پردازنده گرافیکی پیشرفته به دست

آورد). معماری یکپارچه YOLO امکان استنتاج سریعتر را فراهم می کند زیرا تصویر را در یک pass پردازش می کند.

- SSD: SSD همچنین برای سرعت طراحی شده است و می تواند عملکردی در زمان واقعی داشته باشد، اگرچه ممکن است به دلیل نقشه های ویژگی چند مقیاسی کمی کندتر از YOLO باشد. استفاده از نقشه های چندگانه ویژگی برای پیش بینی، مقداری سربار محاسباتی را اضافه می کند، اما همچنان در مقایسه با آشکارسازهای دو مرحله ای کارآمد باقی می ماند.

3. دقت:

- یولو: YOLO روی اشیاء بزرگتر به خوبی عمل می کند و خطاهای پس زمینه کمتری دارد. رویکرد ارزیابی واحد آن به حفظ تعادل بین سرعت و دقت کمک می کند. نسخه های قبلی YOLO با شناسایی اشیاء کوچکتر و مدیریت اشیایی که بسیار نزدیک به یکدیگر هستند، مشکل داشت. YOLOv3 و نسخه های بعدی از این جنبه بهبود یافته اند، اما همچنان ممکن است در تشخیص اجسام کوچک از SSD عقب باشند.
- SSD: SSD به دلیل نقشه های ویژگی چند مقیاسی و جعبه های پیش فرض در تشخیص اشیاء با اندازه های مختلف برتری دارد. به طور کلی در مقایسه با نسخه های قبلی YOLO روی اشیاء کوچکتر عملکرد بهتری دارد. اگرچه SSD دقیق است، ممکن است در برخی سناریوها به سرعت YOLO نباشد. علاوه بر این، SSD به دلیل استفاده گسترده از جعبه های پیش فرض، گاهی اوقات می تواند نتایج کاذب بیشتری تولید کند.

4. سناریوهایی که ممکن است یکی از دیگری بهتر عمل کند:

- یولو:

- برنامه های بلادرنگ: سرعت بالای YOLO آن را برای برنامه های بلادرنگ مانند نظارت تصویری، رانندگی مستقل و تشخیص اشیا زنده در برنامه های تلفن همراه مناسب می کند.

- اشیا بزرگتر: YOLO ممکن است در سناریوهایی که اشیا بزرگتر و متمایزتر هستند، مانند تشخیص عابر پیاده در وسایل نقلیه خودران یا تشخیص وسایل نقلیه بزرگ در نظارت بر ترافیک، بهتر عمل کند.

- SSD:

- اندازه های مختلف اشیا: نقشه های ویژگی چند مقیاسی SSD آن را در مدیریت اندازه های مختلف اشیا در یک تصویر ماهرتر می کند. این باعث می شود آن را برای برنامه هایی مانند تشخیص عمومی اشیا در تصاویر که در آن اشیا با اندازه های مختلف ظاهر می شوند مناسب باشد.

- اشیا کوچک: در سناریوهایی که تشخیص اجسام کوچک حیاتی است، مانند تشخیص حیوانات کوچک یا اشیا در تصاویر پهپاد، SSD ممکن است بهتر از YOLO عمل کند.

ب) focal loss یک شکل اصلاح شده از cross-entropy loss است که برای رفع مشکل عدم تعادل کلاس در وظایف تشخیص اشیا طراحی شده است. ایده کلیدی پشت آن تمرکز بیشتر روی نمونه هایی که طبقه بندی آنها دشوار است و کاهش وزن تخصیص یافته به نمونه های خوب طبقه بندی شده، است. که فرمول آن به شرح زیر است:

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

عدم تعادل کلاس یک مشکل رایج در تشخیص شی است، که در آن تعداد نمونه های پس زمینه (منفی) بسیار بیشتر از نمونه های پیش زمینه (مثبت) است. این عدم تعادل می تواند باعث شود که مدل در طول آموزش توسط کلاس پس زمینه غرق شود و منجر به عملکرد ضعیف در کلاس های اقلیت (پیش زمینه) شود.

این loss این مشکل را به روش های زیر برطرف می کند:

1. نمونه های آسان کاهش وزن: نمونه های آسان (که به درستی با اطمینان بالا طبقه بندی شده اند) به دلیل تعداد زیادشان، علی رغم کمک کمی به بهبود مدل، بر ضرر غالب هستند. این روش تاثیر این نمونه های آسان را با کاهش وزن آنها کاهش می دهد. این به مدل اجازه می دهد تا بیشتر بر روی نمونه های سخت تمرکز کند، که معمولاً کمتر ارائه می شوند اما اطلاعات بیشتری دارند.

2. تمرکز روی مثال های سخت: با تنظیم پارامتر فوکوس یا همان گاما، Focal Loss تضمین می کند که نمونه های طبقه بندی شده اشتباه (نمونه های سخت) سهم بیشتری در از دست دادن کل دارند. این مکانیسم توانایی مدل را برای یادگیری از نمونه های چالش برانگیز بهبود می بخشد و در نتیجه عملکرد کلی آن را در تشخیص های دشوار افزایش می دهد.

3. تعادل مشارکتهای کلاس: گنجاندن ضریب وزنی کلاس خاص یا همان آلفا، بیشتر به متعادل کردن سهم کلاس های مختلف کمک می کند. این امر به ویژه در مجموعه داده هایی با عدم تعادل طبقاتی قابل توجه مفید است و تضمین می کند که در طول آموزش به کلاس های اقلیت اهمیت مناسب داده می شود.

سوال 3: Non-Maximum Suppression (NMS) یک تکنیک پس پردازشی است که در تشخیص اشیا برای انتخاب بهترین جعبه مرزی برای هر شی شناسایی شده

و حذف کادرهای همپوشانی اضافی استفاده می شود. مدل های تشخیص اشیاء، مانند YOLO، SSD اغلب bounding box متعددی را برای یک شی تولید می کنند. این مدل ها می توانند چندین جعبه همپوشانی با امتیازات اطمینان بالا در اطراف یک شی ایجاد کنند. NMS به موارد زیر کمک می کند:

- کاهش افزونگی: NMS با انتخاب مناسب ترین جعبه مرزی و سرکوب دیگران، افزونگی را در نتایج تشخیص کاهش می دهد.
  - بهبود دقت: حذف تشخیص های تکراری به ارائه محلی سازی واضح تر و دقیق تر اشیاء کمک می کند.
  - افزایش وضوح: تضمین می کند که هر شی شناسایی شده توسط یک جعبه محدود نمایش داده می شود و خروجی را تمیزتر و تفسیر را آسان تر می کند.
- مراحل مربوط به اعمال NMS به شرح زیر است:

1. مقداردهی اولیه: با لیستی از جعبه های محدود و امتیازات اطمینان مربوط به آنها شروع میکنیم. ابتدا یک لیست خالی برای نگهداری کادرهای انتخابی نهایی ایجاد میکنیم.
2. مرتب سازی: جعبه های مرزی را بر اساس امتیازات اطمینان آنها به ترتیب نزولی مرتب میکنیم.
3. انتخاب تکراری: کادر محدود با بالاترین امتیاز اطمینان را انتخاب کرده و آن را به لیست کادرهای انتخابی نهایی اضافه میکنیم. سپس این کادر را با سایر کادرهای باقیمانده مقایسه کرده و  $IoU$  را برای هر جفت محاسبه میکنیم.



4. حذف: کادرهایی که دارای  $IoU$  بالا (بالتر از آستانه مشخص شده) هستند را با کادر انتخاب شده حذف میکنیم. اینها اضافی در نظر گرفته می شوند زیرا احتمالاً با یک شی مطابقت دارند.
5. تکرار: این فرآیند را با بالاترین کادر اطمینان بعدی تکرار کرده و تا زمانی که همه کادرها انتخاب یا حذف شوند ادامه میدهیم.
6. خروجی: لیست نهایی شامل کادرهای کراندار غیر همپوشانی است که به بهترین شکل اشیاء شناسایی شده را نشان می دهد.

```
def non_maximum_suppression(bboxes, scores, iou_threshold):
    # Step 1: Sort the bounding boxes by their confidence scores in descending order
    indices = sorted(range(len(scores)), key=lambda i: scores[i], reverse=True)

    selected_indices = []

    while indices:
        # Step 2: Select the box with the highest score
        current_index = indices.pop(0)
        selected_indices.append(current_index)

        # Step 3: Suppress all overlapping boxes with IoU above the threshold
        filtered_indices = []
        for i in indices:
            if compute_iou(bboxes[current_index], bboxes[i]) < iou_threshold:
                filtered_indices.append(i)
        indices = filtered_indices

    return selected_indices
```

تاثیر آستانه های مختلف  $IoU$ : آستانه  $IoU$  تعیین می کند که چه مقدار همپوشانی بین جعبه های محدود کننده مجاز است قبل از اینکه آنها اضافی در نظر گرفته شوند و یکی حذف شود.

- آستانه  $IoU$  پایین (به عنوان مثال، 0.3):

○ سرکوب سخت‌تر: جعبه‌های محدودکننده باید همپوشانی بسیار کمی داشته باشند تا هر دو حفظ شوند. حتی همپوشانی‌های جزئی منجر به سرکوب می‌شود.

○ تشخیص‌های کمتر: این می‌تواند به جعبه‌های محدود نهایی کمتری منجر شود، که ممکن است منجر به گم شدن برخی از اشیاء شود که تشخیص‌های کمی همپوشانی دارند.

○ Reduced Redundancy: در حذف موارد تکراری تهاجمی تر است و احتمال وجود چندین جعبه در اطراف یک شی را کاهش می‌دهد.

● آستانه IoU بالا (به عنوان مثال، 0.7):

○ سرکوب ملایم: جعبه‌های محدودکننده می‌توانند قبل از سرکوب شدن، همپوشانی بیشتری داشته باشند.

○ تشخیص‌های بیشتر: این می‌تواند منجر به حفظ جعبه‌های محدودکننده بیشتر، از جمله موارد بالقوه همپوشانی شود.

○ افزونگی افزایش یافته: در حذف موارد تکراری کمتر تهاجمی است، که ممکن است منجر به ایجاد چندین جعبه محدود در اطراف یک شی شود.

انتخاب آستانه IoU مناسب: آستانه IoU بهینه به کاربرد خاص و ویژگی‌های مجموعه داده بستگی دارد:

● صحنه‌های شلوغ: در سناریوهایی با اشیاء نزدیک بسته، آستانه IoU بالاتر ممکن است برای اطمینان از اینکه اشیاء همپوشانی از دست نمی‌روند مفید باشد.

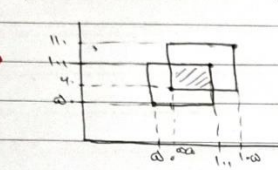
- صحنه‌های پراکنده: برای مجموعه‌های داده با اشیاء به خوبی جدا شده، یک آستانه IOU کمتر ممکن است برای کاهش افزونگی و بهبود وضوح مناسب‌تر باشد.

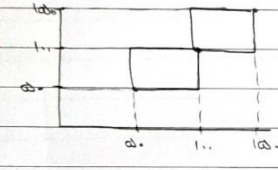
سوال 4:

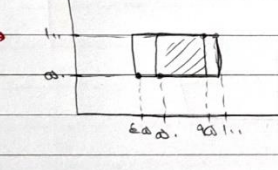
① ابتدا box ها را براساس Confidence Score مرتب می‌کنیم:

1.  $(19, (50, 50, 150, 150)) \rightarrow$  Confidence score بالاتر box
2.  $(18, (50, 40, 150, 150))$
3.  $(17, (50, 100, 150, 150))$
4.  $(14, (50, 50, 90, 90))$

② با استفاده از فرمول IOU محاسب می‌کنیم:

1, 2  $\rightarrow$    $\Rightarrow$  اشتراك  $= (150 - 50) \times (150 - 40) = 1800$   
 $\Rightarrow$  اشتراك  $= 50 \times 50 + 50 \times 50 - 1800 = 3200$   
 $\Rightarrow$  IOU  $= \frac{1800}{3200} = 0.5625$  بسیار خفیف بود  $\Rightarrow 0.5625$

1, 3  $\rightarrow$    $\Rightarrow$  اشتراك  $= 0$  بسیار خفیف نبود  $\Rightarrow 0$

1, 4  $\rightarrow$    $\Rightarrow$  اشتراك  $= (90 - 50) \times 50 = 2000$   
 $\Rightarrow$  اشتراك  $= 50 \times 50 + 50 \times 50 - 2000 = 2750$   
 $\Rightarrow$  IOU  $= \frac{2000}{2750} = 0.727$  بسیار خفیف بود  $\Rightarrow 0.727$

نتیجه: با استفاده از فرمول IOU می‌توانیم تشخیص دهیم که آیا دو box با هم اشتراك دارند یا نه.

1.  $(19, (50, 50, 150, 150))$
2.  $(18, (50, 100, 150, 150))$

سوال 5: برای اینکه اندازه واقعی پارامترها را بدست بیاوریم، باید از فرمول‌های زیر استفاده کنیم:

$$\begin{aligned}x_{\text{center}} &= bx \times \text{image\_width} \\ y_{\text{center}} &= by \times \text{image\_height} \\ \text{width} &= bw \times \text{image\_width} \\ \text{height} &= bh \times \text{image\_height}\end{aligned}$$

حال به ما مقادیر نرمال شده و ورودی عکس داده شده، پس محاسبات را انجام می‌دهیم:

$$x_{\text{center}} = bx * \text{image\_width} = 0.5 * 416 = 208$$

$$y_{\text{center}} = by * \text{image\_height} = 0.6 * 416 = 249.6$$

$$\text{width} = bw * \text{image\_width} = 0.3 * 416 = 124.8$$

$$\text{height} = bh * \text{image\_height} = 0.4 * 416 = 166.4$$

حال با استفاده از این مقادیر می‌توانیم گوشه بالا سمت چپ و پایین سمت راست مستطیل را بدست آوریم:

- top-left corner ( $x_{\text{min}}, y_{\text{min}}$ ):

$$x_{\text{min}} = x_{\text{center}} - \text{width} / 2 = 208 - (124.8 / 2) = 145.6$$

$$y_{\text{min}} = y_{\text{center}} - \text{height} / 2 = 249.6 - (166.4 / 2) = 166.4$$

- bottom-right corner ( $x_{\text{max}}, y_{\text{max}}$ ):

$$x_{\text{max}} = x_{\text{center}} + \text{width} / 2 = 208 + (124.8 / 2) = 270.4$$

$$y_{\text{max}} = y_{\text{center}} + \text{height} / 2 = 249.6 + (166.4 / 2) = 332.8$$

پس مقادیر واقعی برای باکس تشخیص داده شده، ( $145.6, 166.4$ ) و ( $270.4, 332.8$ ) هستند.