



درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

علیرضا حسینی	نام و نام خانوادگی	پرسش ۱
810100125	شماره دانشجویی	
سید مهدی حاجی سید حسین	نام و نام خانوادگی	پرسش ۲
810100118	شماره دانشجویی	
۱۴۰۴.۲.۲۵ (ارسال دو روز تا خیر که تاخیرها نصف حساب شدند)	مهلت ارسال پاسخ	

فهرست

پرسش ۱. سگمنتیشن تصاویر شهری
1.....	1-1. توصیف مدل ارائه شده
3.....	2-۱. آمادهسازی مجموعه داده
4.....	3-۱. بهینهساز، متريک‌ها و تابع هزينه
4.....	4-۱. IoU Score) Intersection over Union Score): .1 تعريف و فرمول:
4.....	4-2. ويزگي‌ها:
5.....	5-۲. Dice Coefficient Score: .2 تعريف و فرمول:
5.....	5-3. ويزگي‌ها:
5.....	5-۴. پيادهسازی مدل ۴-۱
5.....	5-۱. Depthwise Separable Convolution (DSConv).1
6.....	6-۱. Pyramid Pooling Module (PPM) .2
6.....	6-۲. Inverted Residual Block .3
6.....	6-۳. Depthwise Convolution .4
6.....	6-۴. آموزش مدل ۵-۱
7.....	7-۱. ارزیابی مدل ۶-۱
پرسش ۲. تشخیص اشیاء به کمک Oriented R-CNN
9.....	2-۱. بخش اول سوالات نظری (درک مفهومی)
9.....	9-۱. (مثال از مقاله رفرنس [26]) Rotated RPN .1
10.....	10-۱. (مثال از مقاله رفرنس شماره [7]) RoI Transformer .2
10.....	10-۲. (مثال از مقاله رفرنس شماره [37]) Gliding Vertex .3
13.....	13-۱. محدودسازی طبیعی فضای جستجو.....
14.....	2-۲. بخش اول سوالات نظری (اجزای مدل)
19.....	2-۳. بخش سوم سوالات نظری (Rotated RoI Align))
19.....	19-۱. گام‌های Rotated RoIAlign
21.....	21-۱. مثال ساده
22.....	2-۴. بخش چهارم سوالات نظری (عملکرد و کارایی)
23.....	23-۱. ارجاع به نتایج مقاله:
25.....	25-۱. 1. استفاده از Oriented RPN سیک
25.....	25-۲. نمایش Midpoint Offset
25.....	25-۳. 3 . تعداد مناسب proposals
28.....	2-۵. پياده سازی عملی (راه اندازی محیط و آماده سازی دیتابست).....
28.....	28-۱. دیتابست HRSC2016-MS
28.....	28-۲. لیل زنی midpoint offset
30.....	30-۱. چند نمونه از لیل زنی و باکس های آن
33.....	33-۱. مدل ارائه شده در مقاله
36.....	36-۱. ارزیابی مدل و تحلیل نتایج
39.....	39-۱. تحلیل مقایسه ای و پیشنهاد بهبود
39.....	39-۲. مقایسه عملکرد:
40.....	40-۱. تحلیل تفاوت ها با مقاله:

40	تفاوت در داده‌ها:
40 ساده loss function
40 data augmentation: کمبود
40 iteration / epoch: کم بودن
41 پیشنهادات عملی برای بهبود :
41 تکمیل loss head:
41 افزایش و تنوع داده‌ها:
41 افزایش تعداد آموزش:

شکل‌ها

- 4..... شکل ۱ . تصاویری از دادگان استفاده شده در آموزش
- 7..... شکل ۲ . ارزیابی مدل
- 10..... شکل ۳ . فرمول محاسبه ی چهار گوشه برای باکس چرخش یافته
- 11..... شکل ۴ . نمونه ای از midpoint offset
- 12..... شکل ۵ . نمونه ای روش قدیمی oriented box
- 13..... شکل ۶ . نمونه ای از عملکرد بد در اندازه باکس به روش قدیمی oriented box
- 13..... شکل ۷ . نمونه ای از عملکرد خوب در اندازه باکس به روش ارائه شده در مقاله
- 15..... شکل ۸ . معماری Oriented RPN (در شکل با کارد قرمز مشخص گشته)
- 16..... شکل ۹ . فرمول تابع هزینه Oriented RPN
- 18..... شکل ۱۰ . فرمول جزیات مقادیر آفست پیشینی شده
- 20..... شکل ۱۱ . فرمول جزئیات مقدار مختصات که با FM مقایسه میشود
- 23..... شکل ۱۲ . نسبت سرعت به دقت (accuracy) در روی DOTA
- 24..... شکل ۱۳ . دقت (accuracy) مدل های مختلف در روی DOTA
- 24..... شکل ۱۴ . سرعت مدل های مختلف در روی DOTA
- 28..... شکل ۱۵ . مثالی از لیبل گذاری آجکت ها با استفاده از فایل XML
- 29..... شکل ۱۶ . یافتن نقطه مرکز و طول و عرض عکس روتیت شده به همراه زاویه
- 29..... شکل ۱۷ . یافتن چهار نقطه کرنر باکس هر شی
- 29..... شکل ۱۸ . روش محاسبه شش پارامتر اصلی هر باکس چرخنده
- 30..... شکل ۱۹ . یافتن باکس در برگیرنده کل شی، هم راستا با محور های مختصات عمودی و افقی
- 30..... شکل ۲۰ . فرمول نهایی محاسبه آلفا و بتا
- 30..... شکل ۲۱ . نمونه ای از کادر باندینگ باکس چرخیده شده
- 31..... شکل ۲۲ . نمونه ای از کادر باندینگ باکس چرخیده شده
- 31..... شکل ۲۳ . نمونه ای از کادر باندینگ باکس چرخیده شده
- 32..... شکل ۲۴ . نمونه ای از کادر باندینگ باکس چرخیده شده
- 32..... شکل ۲۵ . نمونه ای از کادر باندینگ باکس چرخیده شده
- 33..... شکل ۲۶ . مدل ارائه شده ی Oriented RCNN
- 33..... شکل ۲۷ . قسمت Oriented RPN
- 33..... شکل ۲۸ . قسمت Oriented RCNN Head
- 34..... شکل ۲۹ . اطلاعات در حال آموزش
- 35..... شکل ۳۰ . اطلاعات تابع هزینه در حین آموزش راند سوم ۸ اپیاک (در واقع اپیاک ۲۴ ام در کل)
- 36..... شکل ۳۱ . نمونه اول از خروجی مدل
- 36..... شکل ۳۲ . نمونه دوم از خروجی مدل
- 37..... شکل ۳۳ . نمونه سوم از خروجی مدل
- 37..... شکل ۳۴ . نمونه چهارم از خروجی مدل
- 38..... شکل ۳۵ . نمونه پنجم از خروجی مدل
- 38..... شکل ۳۶ . تابع ارزیابی برای مدل بر روی دادگان تست
- 39..... شکل ۳۷ . دقت مدل بر روی دادگان تست

جدول

پرسش ۱. سگمنتیشن تصاویر شهری

۱-۱. توصیف مدل ارائه شده

یک مدل شبکه عصبی کانولوشنی سریع برای سگمنتیشن معنایی است که برای محاسبه کارآمد بر روی دستگاههای embedded با حافظه کم مناسب است. این مدل با ترکیب ایده‌های فریمورک‌های دو شاخه‌ای و انکودر-دیکودر کلاسیک ارائه شده است.

نحوه عملکرد :

این مدل از یک رویکرد دو شاخه‌ای بهره می‌برد، اما با به اشتراک گذاشتن محاسبات لایه‌های اولیه بین دو شاخه، کارایی را افزایش می‌دهد. بخش "Learning to Downsample" ویژگی‌های سطح پایین را برای چندین شاخه با وضوح متفاوت به طور همزمان محاسبه می‌کند. این شبکه جزئیات فضایی را در وضوح بالا با ویژگی‌های عمیق استخراج شده در وضوح پایین ترکیب می‌کند.

ساختمان معماري Fast-SCNN و نقش هر قسمت در فرآيند سگمنتیشن:

معماري Fast-SCNN از چهار مازول اصلی تشکيل شده است:

۱. **مازول Learning to Downsample**: اين مازول اوليه‌ترین بخش شبکه است و از سه

لایه تشکیل شده است: یک لایه کانولوشن استاندارد (Conv2D) و دو لایه کانولوشن قابل تفکیک عمقی (DSConv). هدف این مازول استخراج ویژگی‌های سطح پایین و کاهش وضوح ورودی به میزان 8 برابر است. این مازول به اشتراک‌گذاری محاسبات ویژگی‌های اولیه بین شاخه‌های وضوح پایین و بالا را امکان‌پذیر می‌کند. تمامی لایه‌های این مازول از گام 2 (stride=2) استفاده می‌کنند.

۲. **استخراج کننده ویژگی سراسری (Global Feature Extractor)**: این مازول برای بدست

آوردن زمینه سراسری برای سگمنتیشن تصویر طراحی شده است. این مازول خروجی inverted مازول Learning to Downsample را دریافت می‌کند و از بلوک‌های (bottleneck residual blocks) استفاده می‌کند که با استفاده از کانولوشن‌های قابل تفکیک عمقی کارآمد ساخته شده‌اند. یک مازول Pyramid Pooling Module - PPM نیز در انتهای این بخش برای جمع‌آوری اطلاعات متعدد مبتنی بر مناطق مختلف اضافه شده است.

۳. **مازول ترکیب ویژگی (Feature Fusion Module)**: این مازول ویژگی‌های بدست آمده از

مازول Learning to Downsample (جزئیات فضایی/مکانی) و استخراج کننده ویژگی

سراسری (دید کلی) را با هم ترکیب می‌کند. برای اطمینان از کارایی، از جمع ساده ویژگی‌ها استفاده می‌شود. جزئیات این مازول در جدول 3 مقاله ارائه شده است.

4. طبقه‌بندی (Classifier): این مازول نهایی مسئول پیش‌بینی نهایی ماسک سگمنتیشن است. در این مازول از دو کانولوشن قابل تفکیک عمقی (DSConv) و یک کانولوشن نقطه‌ای (Conv2D) استفاده می‌شود. اضافه کردن چند لایه بعد از مازول ترکیب ویژگی باعث افزایش دقت می‌شود.

مقایسه مدل‌های انکودر-دیکودر مانند U-Net:

مدل‌های انکودر-دیکودر مانند U-Net چارچوبی رایج برای سگمنتیشن هستند. در این مدل‌ها، بخش انکودر با استفاده از کانولوشن و پولینگ ویژگی‌ها را استخراج کرده و وضوح مکانی را کاهش می‌دهد. بخش دیکودر جزئیات فضایی را از ویژگی‌های با وضوح پایین بازیابی کرده و برچسب‌های اشیاء را پیش‌بینی می‌کند. U-Net به طور خاص از Dense Skip connections برای بهره‌برداری از جزئیات فضایی/مکانی استفاده می‌کند.

تفاوت‌های اصلی:

- ساختار اتصالات پرشی: U-Net از چندین اتصال پرشی در وضوح‌های مختلف و اغلب از بلوک‌های کانولوشنی عمیق استفاده می‌کند. در مقابل، Fast-SCNN به عنوان یک حالت خاص از چارچوب انکودر-دیکودر در نظر گرفته می‌شود، اما تنها از یک اتصال پرشی استفاده می‌کند تا محاسبات و حافظه مورد نیاز را کاهش دهد. این اتصال پرشی در لایه‌های اولیه شبکه قرار داده شده است.
- به اشتراک گذاری محاسبات اولیه: مدل‌های دو شاخه‌ای مرسوم و برخی مدل‌های انکودر-دیکودر، محاسبات اولیه را بین شاخه‌های مختلف به اشتراک نمی‌گذارند. Fast-SCNN با معرفی مازول "Learning to Downsample" محاسبات لایه‌های اولیه را بین دو شاخه به اشتراک می‌گذارد.
- کارایی و سرعت: Fast-SCNN به طور خاص برای دستیابی به سرعت بالا و عملکرد بلادرنگ (Real-Time) و بالاتر از آن طراحی شده است. این مدل در مقایسه با برخی مدل‌های انکودر-دیکودر و دو شاخه‌ای دیگر، سرعت (fps) بالاتری را ارائه می‌دهد.
- ظرفیت مدل: Fast-SCNN با ظرفیت کم (تعداد پارامترهای کمتر) طراحی شده است تا امكان اجرا بر روی دستگاه‌های تعبیه‌شده (embedded) با حافظه کم فراهم شود.

۲-۱. آماده سازی مجموعه داده

دیتاست CamVid استفاده شده توسط ما از [این لینک](#) موجود در kaggle استفاده شده که ممکن از در جزئیات تفاوت های کمی داشته باشد اما با توجه به پاسخ تی ای مربوطه دلیلی بر عدم استفاده از آن وجود نداشت.

در دیتاست استفاده شده ۳۶۹ تصویر برای train، تعداد ۱۰۰ تصویر برای validation و تعداد ۲۳۲ تصویر برای test وجود داشت. در فایل ژوپیتر گزارش دقیقی از درصد لیبل ها و آمارهای دقیق دیگر موجود است.



شکل ۱ . تصاویری از دادگان استقاده شده در آموزش

۱-۳. بهینه‌ساز، متريک‌ها و تابع هزينه

اين دو متريک از معيارهای کليدي در ارزیابی مدل‌های یادگيري ماشین، بهويژه در تسكهای تقسيم‌بندی تصوير (Image Segmentation)، هستند.

:**(IoU Score) Intersection over Union Score .1**

IoU يك معیار رایج برای سنجش عملکرد مدل‌های تقسیم‌بندی معنایی تصویر است. این معیار میزان همپوشانی بین پیش‌بینی‌های مدل و داده‌های واقعی (Ground Truth) را اندازه‌گیری می‌کند.

تعريف و فرمول:

- IoU نسبت تعداد پیکسل‌های مشترک (تقاطع) بین پیش‌بینی و واقعیت به کل پیکسل‌های موجود در پیش‌بینی و واقعیت (اجتماع) است.
- فرمول IoU برای يك کلاس خاص به اين صورت است:

$$IoU = \frac{TP}{TP + FP + FN}$$

كه در آن:

- **True Positive (TP)**: پیکسل‌هایی که به درستی به عنوان کلاس مورد نظر پیش‌بینی شده‌اند.
- **False Positive (FP)**: پیکسل‌هایی که به اشتباه به عنوان کلاس مورد نظر هستند اما پیش‌بینی شده‌اند.
- **False Negative (FN)**: پیکسل‌هایی که متعلق به کلاس مورد نظر هستند اما به اشتباه به کلاس دیگری اختصاص داده شده‌اند.

ویژگی‌ها:

- برای وظایف چندکلاسی، معمولاً **Mean IoU** محاسبه می‌شود که میانگین IoU برای تمام کلاس‌ها است.
- مقدار IoU بین ۰ و ۱ است؛ هرچه مقدار به ۱ نزدیک‌تر باشد، همپوشانی بیشتر و عملکرد مدل بهتر است.

:Dice Coefficient Score .2

معیاری مشابه IoU است که شباهت بین دو مجموعه (پیش‌بینی و واقعیت) را اندازه‌گیری می‌کند و به‌ویژه در وظایف تقسیم‌بندی تصویر کاربرد دارد. این معیار گاهی به نام "F1-Score" نیز شناخته می‌شود.

تعريف و فرمول:

- Dice Coefficient نسبت دو برابر تقاطع به مجموع مناطق پیش‌بینی‌شده و واقعی است.
- فرمول آن برای یک کلاس خاص به این صورت است:

$$Dice = \frac{2 \times TP}{(2 \times TP) + FP + FN}$$

که در آن TP ، FP و FN همان معانی ذکر شده در IoU را دارند.

ویژگی‌ها:

- مقدار Dice نیز بین 0 و 1 است و هرچه به 1 نزدیک‌تر باشد، نشان‌دهنده عملکرد بهتر مدل است.
- به دلیل ضریب 2 در صورت کسر، معمولاً مقادیر بالاتری نسبت به IoU تولید می‌کند، به‌ویژه در مواردی که همپوشانی کم است.
- این معیار بیشتر بر دقت تمرکز دارد و به عنوان مکملی برای IoU استفاده می‌شود.

۱۴-۱. پیاده‌سازی مدل

Depthwise Separable Convolution (DSConv).1

- این بلاک از دو بخش اصلی تشکیل شده است:
 - یک نوع convolution که به صورت جداگانه روی هر کanal ورودی اعمال می‌شود و تعداد پارامترها و محاسبات را کاهش می‌دهد.
 - یک 1×1 convolution که ویژگی‌های استخراج شده توسط depthwise convolution را ترکیب می‌کند.
- کاربرد: در مدل FastSCNN، این بلاک برای کاهش وضوح تصویر و استخراج ویژگی‌ها در مراحل اولیه استفاده می‌شود.

Pyramid Pooling Module (PPM) .2

- این مازول برای استخراج ویژگی‌های چند مقیاسی طراحی شده است. با استفاده از pooling در اندازه‌های مختلف (مثلاً 1، 2، 3، 6) و سپس اعمال convolution و normalization، ویژگی‌هایی در مقیاس‌های متفاوت تولید می‌شوند. این ویژگی‌ها با ویژگی‌های اصلی ترکیب می‌شوند تا اطلاعات زمینه‌ای بیشتری فراهم کنند.
- کاربرد: در بخش Global Feature Extractor استفاده می‌شود و به بهبود درک کل تصویر کمک می‌کند.

Inverted Residual Block .3

- این بلاک از معماری MobileNetV2 الهام گرفته شده و شامل سه مرحله است:
 - یک 1×1 convolution که تعداد کانال‌ها را افزایش می‌دهد.
 - برای استخراج ویژگی‌ها بدون تغییر تعداد کانال‌ها.
 - یک 1×1 convolution که تعداد کانال‌ها را به مقدار دلخواه کاهش می‌دهد.
- **ویژگی خاص:** از اتصالات residual استفاده می‌کند تا جریان اطلاعات و آموزش شبکه را بهبود دهد.
- کاربرد: به صورت متوالی در بخش Global Feature Extractor استفاده می‌شود.

Depthwise Convolution .4

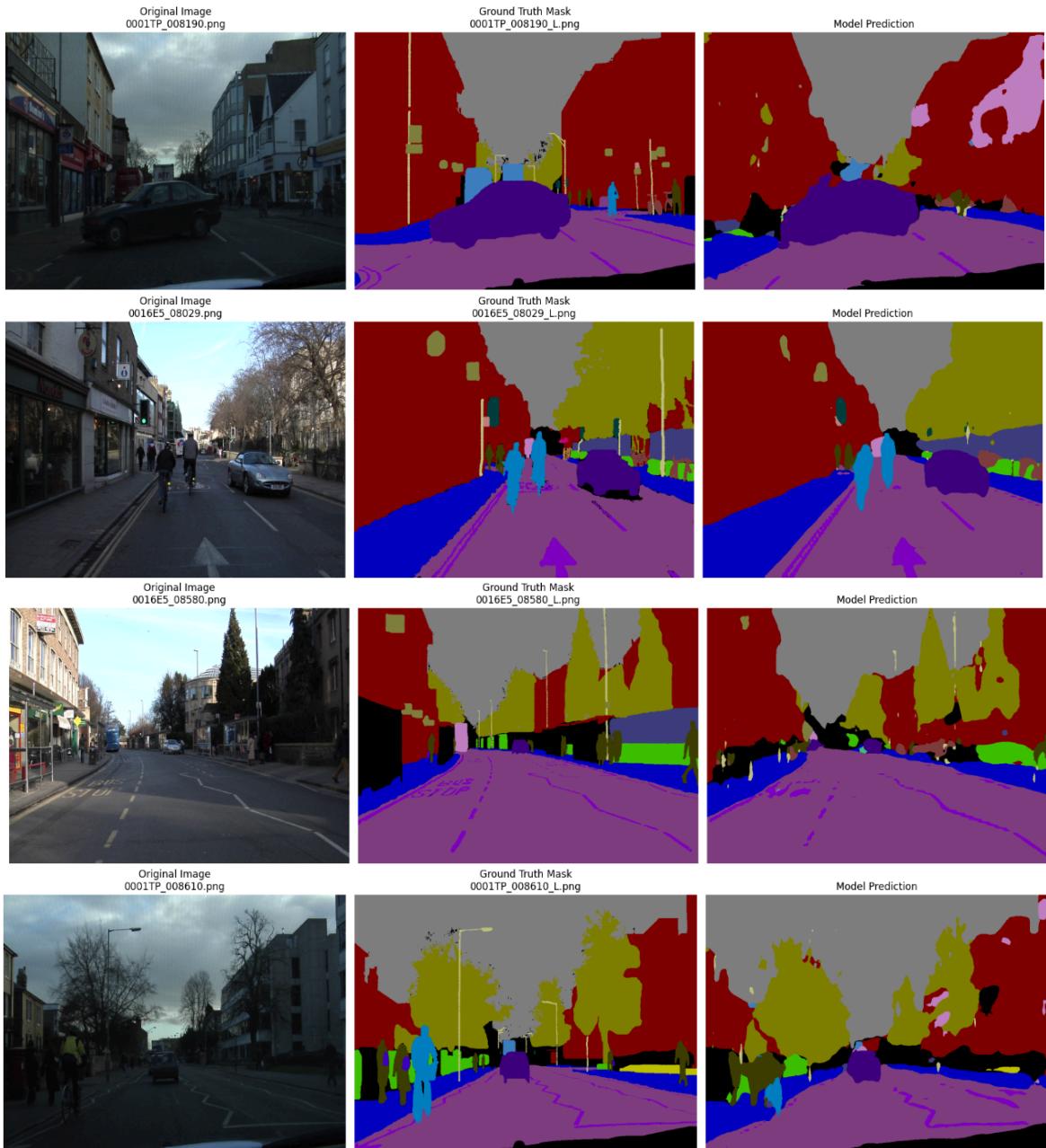
- نوعی convolution است که هر کانال ورودی را به صورت جداگانه با یک فیلتر پردازش می‌کند. این روش تعداد پارامترها و محاسبات را در مقایسه با convolution استاندارد به طور قابل توجهی کاهش می‌دهد.
- کاربرد: به عنوان بخشی از بلاک‌های Inverted Residual Block و DSConv در مدل FastSCNN استفاده می‌شود.

۱-۵. آموزش مدل

جزییات آموزش مدل و آمار و نمودارهای مدل در هر EPOCH در فایل زوپیتر همراه گزارش مشخص است. همچنین اکثر پارامترها داینامیک بوده و بارها عملیات آموزش انجام شده تا به بهترین پارامترهای موجود دست پیدا کنیم. در نتیجه خروجی باقی‌مانده بهترین نتایج ممکن با دیتابست فعلی است. ممکن است تفاوت‌های کوچک دیتابست استفاده شده با دیتابست پیشنهاد شده باعث تفاوت در عملکرد نهایی شده باشد.

۱-۶. ارزیابی مدل

نتایج تست بهترین مدل آموزش دیده شده در متريک‌های متفاوت در فایل ژوپیتر وجود دارد. ۴ نمونه از تصاویر ارزیابی، ماسک واقعی و ماسک پيش‌بينی شده توسط مدل در زير آورده شده است. بقیه تصاویر (۶ تصویر اضافه ديگر) در فایل ژوپیتر موجود است :



شکل ۲. ارزیابی مدل

عملکرد مدل در تشخیص کلی اشیا عموماً بد نبوده است اما در تشخیص مرز اشیا به اندازه کافی خوب عمل نمی‌کند که می‌تواند در نتیجه تعداد کم داده ورودی باشد. همچنین در تصاویری که نور آن به نسبت کمتر از بقیه بوده ضعیفتر عمل کرده که شاید بتواند با Augmentation بیشتر این مورد را بهتر نمود.

پرسش ۲. تشخیص اشیاء به کمک Oriented R-CNN

۱- بخش اول سوالات نظری (درک مفهومی)

- انگیزه اصلی از تولید توسعه Oriented R-CNN را توضیح دهید . این مدل چه محدودیت هایی از روش های قبلی را برطرف میکند ؟ مثال هایی ارائه کنید .

انگیزه اصلی از توسعه Oriented R-CNN رفع محدودیت‌ها و مشکلات موجود در روش‌های قبلی تشخیص اشیاء چرخیده (oriented object detection) بوده است. در ادامه دلایل اصلی توسعه این مدل و مشکلاتی که برطرف می‌کند را همراه با مثال توضیح می‌دهم:

انیگزه اصلی:

روش‌های قبلی برای تشخیص اشیاء با زاویه معمولاً:

- محاسبات سنگینی داشتند (مثل استفاده از هزاران anchor چرخیده با زوایای مختلف)
- پیچیدگی بالا در معماری داشتند.
- باعث کاهش سرعت تشخیص و افزایش مصرف حافظه می‌شدند.

بنابراین هدف Oriented R-CNN این بود که:

- تشخیص دقیق و سریع اشیاء چرخیده فراهم کند
- بدون نیاز به anchor های چرخیده یا عملیات پیچیده
- و همچنان دقیق روش‌های پیشرفته قبلی را حفظ یا حتی بهبود دهد.

محدودیت‌های روش‌های قبلی که برطرف شده‌اند:

(مثال از مقاله رفرانس [26] Rotated RPN .1

- از تعداد زیادی anchor با زوایای مختلف (مثلاً 54 در هر موقعیت) استفاده می‌کرد.
- باعث مصرف زیاد حافظه و کاهش سرعت می‌شد.

اما Oriented R-CNN فقط با anchor های افقی کار می‌کند و از یک نمایش 6 پارامتری برای تبدیل آن‌ها به جعبه‌های چرخیده استفاده می‌کند بنابراین مصرف حافظه بسیار کمتری دارد.

(مثال از مقاله رفرانس شماره [7] RoI Transformer .2

- ابتدا از RPN برای تولید جعبه افقی استفاده می کرد، سپس آنها را به جعبه های چرخیده تبدیل می کرد با استفاده از RoI Align و چند مرحله رگرسیون.
- محاسبات پیچیده و کند بود.

اما Oriented R-CNN در Oriented RPN مستقیماً جعبه چرخیده تولید می کند. (بدون نیاز به چند مرحله و ساده تر شدن شبکه.)

(مثال از مقاله رفرانس شماره [37] Gliding Vertex .3

- تنها رئوس جعبه چرخیده را پیش بینی می کرد.
- همچنان از جعبه های افقی برای ناحیه بندی اولیه استفاده می کرد.

اما Rotated RoIAlign از ابتدا تا انتهایا جعبه چرخیده کار می کند و از استفاده می کند (ویژگی ها بهتر با شیء هم راست است).

- مزایای استفاده از نمایش midpoint offset نسبت به نمایش های سنتی جعبه محدود کننده را توضیح دهید . مثالی برای روشن شدن توضیحات ارائه کنید.

ابتدا یک توضیح مختصری راجع به این مدل نمایش میدهیم :

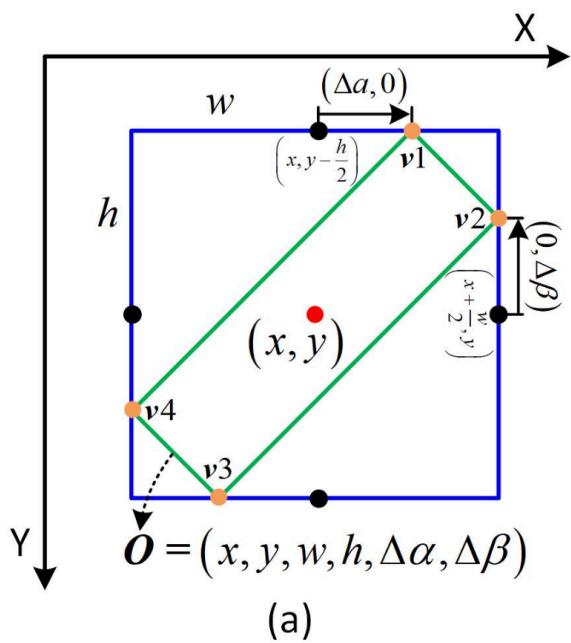
در این مدل هر باکس چرخش یافته (oriented bounding box) را با استفاده از ۶ پارامتر نمایش میدهیم ، این ۶ پارامتر شامل مختصات مرکز ، طول و عرض و همچنین اندازه چرخش روی محور افقی و اندازه چرخش روی محور عمودی است .

فرمول توضیح داده شده در مقاله که برای به دست اوردن چهار گوشه باکس هستند به فرم زیر است : (x و y نشان دهنده مختصات مرکز ، h و w نشان دهنده ارتفاع و عرض و $\Delta\alpha$ ، $\Delta\beta$ نشان دهنده اندازه چرخش هستند)

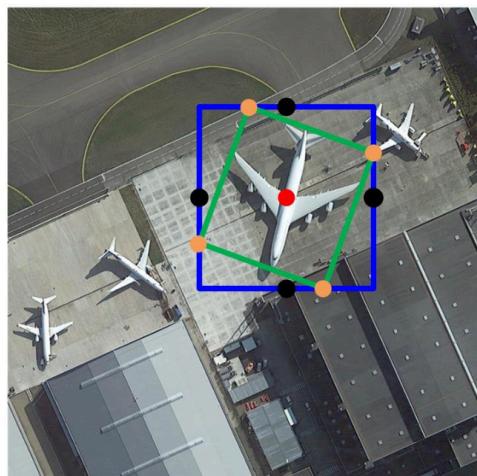
$$\left\{ \begin{array}{l} v1 = (x, y - h/2) + (\Delta\alpha, 0) \\ v2 = (x + w/2, y) + (0, \Delta\beta) \\ v3 = (x, y + h/2) + (-\Delta\alpha, 0) \\ v4 = (x - w/2, y) + (0, -\Delta\beta) \end{array} \right.$$

شکل ۳ . فرمول محاسبه ی چهار گوشه برای باکس چرخش یافته

به عنوان مثال در شکل زیر می توانید یک نمونه که در مقاله ارائه شده را ببینید :



(a)



(b)

شکل ۴. نمونه ای از midpoint offset

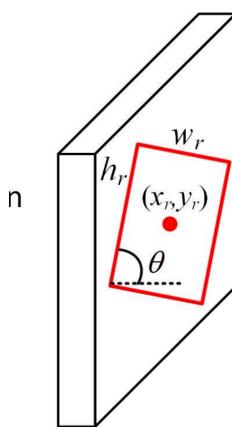
مزایای استفاده از Midpoint Offset

1. سادگی عددی (Numerical Stability)

نمایش زاویه θ در روش‌های سنتی می‌تواند ناپایدار باشد (مثلاً تغییرات کوچک زاویه منجر به تغییرات زیاد در پیش‌بینی می‌شود).

در واقع Midpoint Offset از زاویه اجتناب می‌کند و با تغییرات تدریجی در $\Delta\alpha$, $\Delta\beta$ کنترل بهتری دارد.

مثال :



شکل ۵ . نمونه‌ای روش قدیمی oriented box

2. کمتر بودن پارامتر نسبت به روش گوشه‌ای

نمایش گوشه‌ای نیاز به ۸ عدد دارد ($x_1, y_1, \dots, x_4, y_4$) اما با استفاده از این مدل به ۶ پارامتر نیاز داریم که این باعث می‌شود مدل کمتر احتمالاً در آینده overfit کند .

3. محدودسازی طبیعی فضای جستجو

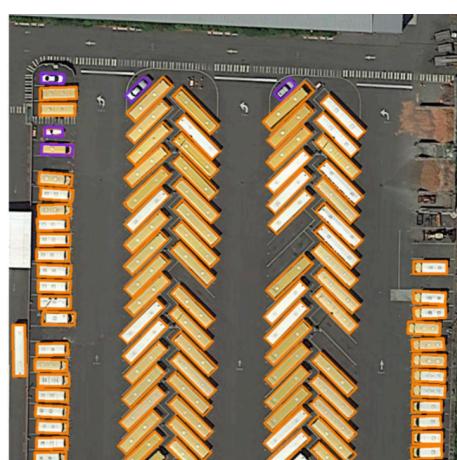
چون $\Delta\alpha$, $\Delta\beta$ بر اساس ابعاد w و h محدود هستند، پیش‌بینی‌ها در فضای معقول باقی می‌مانند. به مدل کمک می‌کند که پیش‌بینی‌های غیرواقعی انجام ندهد.

مثال :

همانطور که در شکل زیر مشاهده می‌شود روش‌های قدیمی، در بین proposal‌های ارائه شده گاها باکس‌های غیر واقعی پدیدار می‌شود که این به دلیل این است که پارامترهای چرخشی در ابعاد h , W محدود نیستند اما در روش ارائه شده در مقاله تمرین، همان طور که در شکل بعدی مشخص است، با توجه به ابعاد W , h چرخش ما تنظیم می‌شود که باعث ساخت باکس‌های واقعی‌تر و هماهنگ‌تر با اندازه می‌شود.



شکل ۶ . نمونه‌ای از عملکرد بد در اندازه باکس به روش قدیمی oriented box



شکل ۷ . نمونه‌ای از عملکرد خوب در اندازه باکس به روش ارائه شده در مقاله

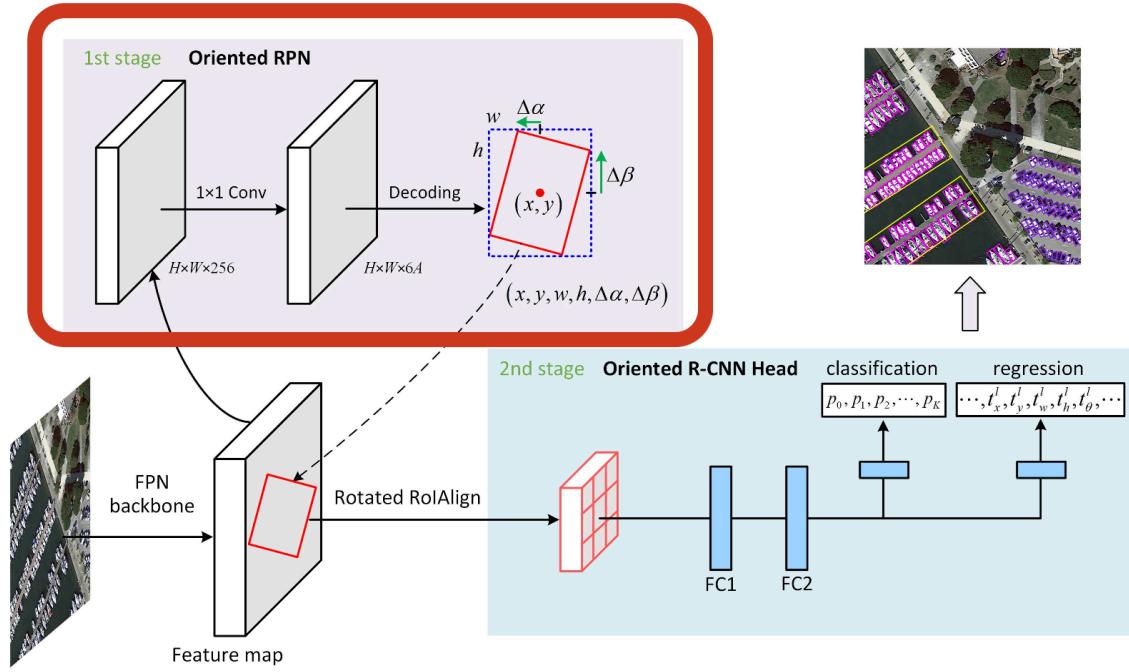
۲-۲. بخش اول سوالات نظری (اجزای مدل)

- معماری oriented RPN را شرح دهید و تفاوت آن با RPN سنتی را توضیح دهید.

در این معماری ورودی های Feature Pyramid Pooling از طریق Feature Map در ۵ سطح مقیاس (P6 تا P2) گرفته می‌شوند که این این لایه‌ها اشیاء در اندازه‌های مختلف را پوشش می‌دهند. سپس در هر مکان مکانی (spatial location) سه anchor افقی با نسبت‌های ابعاد ۱:۱، ۱:۲ و ۲:۱ قرار می‌گیرد. یک هدر سبک شامل یک لایه کانولوشن 3×3 و دو شاخه‌ی موازی (یکی برای طبقه‌بندی شیء و دیگری برای رگرسیون) روی این ویژگی‌ها سوار می‌شود. شاخه رگرسیون به جای چهار پارامتر (x, y, w, h , $\Delta\alpha, \Delta\beta$) شش پارامتر ($x, y, w, h, \Delta\alpha, \Delta\beta$) را پیش‌بینی می‌کند که نمایش Midpoint Offset نامیده می‌شود؛ $\Delta\alpha$ و $\Delta\beta$ به عنوان جابجایی نسبت به نقاط میانی بالا/پایین و چپ/راست جعبه افقی عمل کرده و چرخش را به سادگی مدل می‌کنند. در نهایت با تبدیل این شش پارامتر به مختصات چهار راس جعبه، پیشنهادهای چرخیده با دقت بالا و بدون نیاز به anchor های زاویه‌دار تولید می‌شوند. این رویکرد ضمن حفظ سرعت و سادگی RPN، بهره‌وری محاسباتی را افزایش داده و برای اشیاء با جهت‌های دلخواه نتایج بسیار موثرتری ارائه می‌دهد.

تفاوت آن با مدل قدیمی :

در مقابل RPN سنتی که تنها چهار پارامتر (x, y, w, h) برای تولید جعبه‌های افقی پیش‌بینی می‌کند، Oriented RPN شش پارامتر ($x, y, w, h, \Delta\alpha, \Delta\beta$) را رگرسیون می‌کند تا چرخش اشیاء را هم در نظر بگیرد. در RPN قدیمی در هر نقطه از نقشه ویژگی فقط anchorهای افقی با ابعاد مختلف قرار می‌گرفت و سپس با اعمال رگرسیون همین چهار پارامتر به جعبه نهایی می‌رسیدیم؛ اما در Oriented RPN، همان anchorهای افقی اولیه با نسبت‌های ابعاد ۱:۲، ۱:۱ و ۲:۱ استفاده می‌شوند و شاخه رگرسیون به کمک نمایش «Midpoint Offset» با دو جابه‌جایی $\Delta\alpha$ و $\Delta\beta$ ، گوشه‌های جعبه را به سمت بالا/پایین و چپ/راست می‌لغزاند تا زاویه چرخش را مدل کند. در نتیجه دیگر نیازی به تعریف دههای anchor مورب با زوایای مختلف نیست و شبکه با همان پیچیدگی محاسباتی RPN معمولی می‌تواند پیشنهادهای چرخیده با دقت بالا تولید کند.



شکل ۸ . معماری Oriented RPN (در شکل با کارد قرمز مشخص گشته)

- نحوه فرمول بندی تابع هزینه (Loss Function) را در Oriented RPN بیان کنید و هر یک از اجزای این تابع را به طور واضح تعریف و هدف آنها را شرح دهید.

در مقاله ذکر شده است که ابتدای کار ، به هر anchor یک لیبل میدادند (که در مقاله آن را با علامت P^* تعریف کرده اند .

در صورتی anchor در لیبل 1 قرار دارد که :

- که این anchor یک Intersection-over-Union (IoU) با هر Ground Truth box (با ای ای) با مقدار بیشتر از ۰.۷ داشته باشد .

یا

- این Anchor بیشترین Gournd Truth box را با IoU Overlap داشته باشد و مقدار آن بیشتر از ۰.۳ باشد .

در غیر این صورت اگر هر anchor ای عضو هیچ یک از این لیبل ها نباشد در نظر گرفته نمیشود .

در ادامه بر اساس فرمول بندی موجود در مقاله، تابع هزینه‌ی Oriented RPN به شکل زیر تعریف می‌شود:

$$L_1 = \frac{1}{N} \sum_{i=1}^N F_{cls}(p_i, p_i^*) + \frac{1}{N} p_i^* \sum_{i=1}^N F_{reg}(\delta_i, t_i^*) \quad (3)$$

شکل ۹ . فرمول تابع هزینه Oriented RPN

که در آن:

N . 1

تعداد کل anchor های نمونه‌برداری شده در یک mini-batch (به طور پیش‌فرض N=256) است.

P i .2

خروجی شاخه‌ی طبقه‌بندی برای α -امین anchor، یعنی احتمال «دارای شیء بودن» آن.

P i* .3

لیبل واقعی آن anchor بر اساس شرایط زیر:

$p_{i^*} = 1$ (مثبت) اگر: ○

.1. IoU آن با هر Ground-Truth Box از 0.7 بیشتر باشد، یا

.2. بالاترین IoU را با یک Ground-Truth Box داشته باشد و مقدار IoU اش از 0.3 بیشتر باشد.

$p_{i^*} = 0$ (منفی) اگر IoU آن با همه‌ی Ground-Truth Box‌ها کمتر از 0.3 باشد.

○ در غیر این صورت (یعنی نه مثبت و نه منفی) نادیده گرفته می‌شود.

F_{cls} .4

تابع خطای طبقه‌بندی (Cross-Entropy) که شبکه را قادر می‌کند بین «شیء» و «پس‌زمینه» تمایز قائل شود.

$\delta_i = \delta x, \delta y, \delta w, \delta h, \delta \alpha, \delta \beta$.5

بردار آفست پیش‌بینی‌شده توسط شاخه‌ی رگرسیون برای α -امین anchor. چهار مؤلفه‌ی اول مربوط به ترجمه و مقیاس جعبه خارجی (همانند RPN سنتی) و دو مؤلفه‌ی آخر یعنی، $\delta \alpha, \delta \beta$ مربوط به جابجایی midpoint جهت کنترل زاویه چرخش هستند.

$*t_i = t_x^*, t_y^*, t_w^*, t_h^*, t_\alpha^*, t_\beta$.6

بردار آفست هدف که از Ground-Truth استخراج می‌شود. روابط پارامترها به صورت زیر

است:

$$\left\{ \begin{array}{ll} \delta_\alpha = \Delta\alpha/w, & \delta_\beta = \Delta\beta/h \\ \delta_w = \log(w/w_a), & \delta_h = \log(h/h_a) \\ \delta_x = (x - x_a)/w_a, & \delta_y = (y - y_a)/h_a \\ t_\alpha^* = \Delta\alpha_g/w_g, & t_\beta^* = \Delta\beta_g/h_g \\ t_w^* = \log(w_g/w_a), & t_h^* = \log(h_g/h_a) \\ t_x^* = (x_g - x_a)/w_a, & t_y^* = (x_g - x_a)/h_a \end{array} \right.$$

شکل ۱۰ . فرمول جزیات مقادیر آفست پیش‌بینی شده

$$F_{Reg} .7$$

تابع خطای رگرسیون (SmoothL1) که شبکه را به یادگیری دقیق آفست‌های تعریف‌شده و ادار می‌کند تا anchor افقی اولیه را به یک جعبه‌ی چرخیده و ادار به انطباق کند.

- هدف بخش طبقه‌بندی، این است که RPN بتواند anchor های مربوط به اشیاء واقعی را از پس زمینه تشخیص دهد و در نتیجه فقط روی نمونه‌های مفید رگرسیون انجام شود.
- هدف بخش رگرسیون آموزش شبکه برای جابجایی و مقیاس‌دهی anchor ها و در نهایت چرشش آنها از طریق نمایش Midpoint Offset است، به‌طوری که پیشنهادهای نهایی جعبه‌های چرخیده با دقت بالا تولید شوند.

۲-۳. بخش سوم سوالات نظری (Rotated RoI Align)

- هدف از Rotated RoI Align چیست؟ گام به گام توضیح دهید که چگونه این عملیات انجام میشود و یک مثال ارائه کنید.

هدف استخراج ویژگی‌های مکانی-چرخیده (rotation-invariant) از هر پیشنهاد (proposal) چرخیده است، به طوری که شبکه بتواند به دقت روی خود شیء (و نه بخشی از پس زمینه) تمرکز کند. در ادامه گام‌های انجام این عملیات را شرح می‌دهم و با یک مثال روشن می‌کنم.

گام‌های Rotated RoIAlign

۱. دریافت پروپوزال چرخیده

- از Oriented RPN یک پیشنهاد به صورت یک چندضلعی چهارگوش (parallelogram) یا به طور معادل با چهار راس دریافت می‌شود.

۲. تبدیل به مستطیل چرخیده

برای ساده‌سازی نمونه‌برداری، کوتاهترین قطر چندضلعی را تا طول قطر بلند آن امتداد می‌دهیم و در نتیجه یک مستطیل دارای زاویه θ می‌سازیم.

- این مستطیل با پارامترهای (x, y, w, h, θ) نمایش داده می‌شود که (y) مرکز، w, h ابعاد، و θ زاویه نسبت به محور افقی است.

۳. مقیاس ویژگی (Feature Map Scaling)

- از آنجا که Feature Map نسبت به تصویر ورودی کوچکتر است، مختصات و ابعاد مستطیل باید به آن مقیاس شوند:

$$\begin{cases} w_r = w/s, & h_r = h/s \\ x_r = \lfloor x/s \rfloor, & y_r = \lfloor y/s \rfloor \end{cases}$$

شکل ۱۱ . فرمول جزئیات مقدار مختصات که با FM مقایسه میشود

که s در واقع سطح متناظر در FPN است.

۴. تقسیم به شبکه ثابت (Grid)

- مستطیل چرخیده را به یک شبکه $M * M$ تقسیم میکنیم.
- هر سلول از این شبکه معادل یک موقعیت کوچک در منطقه چرخیده است.

۵. نمونهبرداری چرخشی (Rotated Sampling)

- برای هر سلول (i,j) ، مختصات نقطه‌های نمونه (یا یک ناحیه کوچک) را با استفاده از ماتریس چرخش θ به مختصات Feature Map میآوریم.
- سپس مقدار ویژگی آن نقاط را با interpolation میخوانیم.

۶. تجمع و خروجی

مقادیر نمونه شده در هر سلول میانگین گرفته میشوند تا یک ماتریس ویژگی $M * M * C$ به دست آید. که C مقدار کanal است.

- این ماتریس، ویژگی‌های چرخیده و همراستا با شیء را در اختیار HEAD تشخیص می‌گذارد.

مثال ساده

تصور کنید تصویری با رزولوشن 1024×1024 داریم که روی آن یک هوایپیما با زاویه 30° درجه نسبت به افق نشسته است. Oriented RPN یک پیشنهاد چرخیده چهارضلعی اطراف هوایپیما می‌دهد:

1. چهار راس پیشنهاد:

$$v1=(600,300), v2=(700,350), v3=(650,450), v4=(550,400)$$

2. تبدیل به مستطیل چرخیده:

تشخیص می‌دهیم مستطیل با

$$(x=625, y=375, w=150, h=200, \theta=30 \text{ Degree}).$$

3. مقیاس به Feature Map فرض کنید که استراید ما ۴ است :

$$(xr=156, yr=93, wr=37.5, hr=50)$$

4. تقسیم به شبکه 7×7

5. برای هر سلو، نقاط نمونه را روی Feature Map می‌چرخانیم و مقدار ویژگی می‌خوانیم.

6. خروجی: یک بلوک $7 \times 7 \times C$ از ویژگی‌های هم‌راستا با هوایپیما.

با این کار، HEAD شبکه دقیقاً با ویژگی‌هایی سروکار دارد که دقیقاً روی شیء چرخیده متمرکز هستند، نه بخش‌های پس‌زمینه یا نواحی برش‌خورده غلط.

- مشکلات احتمالی در صورت عدم استفاده از **Rotated RoI Align** را توضیح دهید ؟
برای استدلال خود مثال یا توجیه نظری ارائه کنید .

در صورت حذف **Rotated RoI Align**, دو مشکل اصلی پیش می آید:

1. عدم هم راستی ویژگی و شیء

اگر از **RoIAlign** افقی استفاده شود، برای یک شیء مورب بخش زیادی از پس زمینه در شبکه نمونه برداری می شود و بخش های کلیدی شیء تار می مانند که نتیجتاً دقت طبقه بندی و رگرسیون شدیداً افت می کند.

2. کاهش دقت در اشیاء کوچک (مثالاً متراکم اما کوچک)

در تصاویر هوایی با اشیاء ریز یا فشرده (مثل خودروهای پارک شده در هم)، باعث می شود نمونه ویژگی «همسایه ها» را بگیرد نه خود شیء که نتیجتاً نرخ خطأ و **False Positive** بالا می رود.

توجیه نظری (برداشت ما از سوال شما این بود که استلالی منطقی برای دو مشکل بالا ارائه بدیم) : بردار ویژگی استخراج شده از یک ناحیه افقی روی شیء چرخیده، ترکیبی از پس زمینه و شیء است؛ اما **Rotated RoIAlign** تضمین می کند که شبکه دقیقاً روی خود شیء و بدون زمینه اضافی پردازش کند

۴-۲. بخش چهارم سوالات نظری (عملکرد و کارایی)

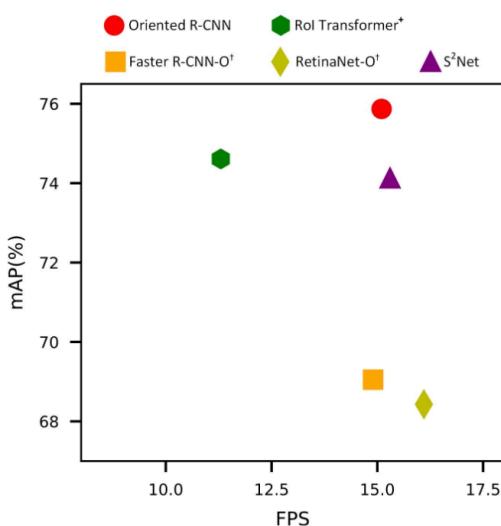
- توضیح دهید که **Oriented R-CNN** چگونه به دقت و کارایی بالا دست پیدا می کند . به طور مشخص به آزمایش ها و نتایج ارائه شده در مقاله ارجاع دهید .

Oriented R-CNN با طراحی ساده و مؤثر خود به دقت و کارایی بالا دست یافته است. این مدل با معرفی **Oriented RPN** سبک و استفاده از نمایش **Midpoint Offset**، جعبه های چرخیده را مستقیماً و با هزینه محاسباتی ناچیز تولید می کند. همچنین با استفاده از **Rotated RoI Align** ویژگی ها دقیقاً از ناحیه هم راستا با شیء استخراج می شوند که باعث افزایش دقت تشخیص می شود.

ارجاع به نتایج مقاله:

- در داده mAP %75.87 رسیده DOTA، Oriented R-CNN با دقت ResNet-50-FPN است که از تمام روش‌های دو مرحله‌ای و تک مرحله‌ای مقایسه شده در جدول ۲ مقاله بهتر است.
- در داده HRSC2016، این مدل با ResNet-101-FPN mAP %97.60 رسیده است که بالاترین مقدار در جدول ۳ مقاله است.
- از نظر سرعت، در جدول ۴، این مدل با FPS 15.1 روی RTX 2080Ti اجرا می‌شود که سرعتی نزدیک به روش‌های تک مرحله‌ای دارد ولی با دقت بالاتر.

بنابراین، Oriented R-CNN با ترکیب معماری بهینه، نمایش جدید و بدون نیاز به anchorهای چرخیده، همزمان دقت بالا و سرعت قابل قبول ارائه می‌دهد.



شکل ۱۲ . نسبت سرعت به دقت (accuracy) در روی DOTA

Method	Backbone	mAP(07)	mAP(12)
PIoU [4]	DLA-34	89.20	-
DRN [28]	H-34	-	92.70
R3Det [39]	R-101-FPN	89.26	96.01
DAL [27]	R-101-FPN	89.77	-
S ² ANet [12]	R-101-FPN	90.17	95.01
Rotated RPN [26]	R-101	79.08	85.64
R2CNN [17]	R-101	73.07	79.73
RoI Transformer [7]	R-101-FPN	86.20	-
Gliding Vertex [37]	R-101-FPN	88.20	-
CenterMap-Net [35]	R-50-FPN	-	92.80
Oriented R-CNN	R-50-FPN	90.40	96.50
Oriented R-CNN	R-101-FPN	90.50	97.60

شکل ۱۳ . دقت (accuracy) مدل های مختلف در روی DOTA

Method	Framework	FPS	mAP
RetinaNet-O [†]	One-stage	16.1	68.43
S ² ANet [12]	One-stage	15.3	74.12
Faster R-CNN-O [†]	Two-stage	14.9	69.05
RoI Transformer ⁺	Two-stage	11.3	74.61
Oriented R-CNN	Two-stage	15.1	75.87

شکل ۱۴ . سرعت مدل های مختلف در روی DOTA

- عوامل مؤثر در کارایی محاسباتی Oriented R-CNN را توضیح دهید و نقش هر یک از این عوامل را مشخص نمایید.

در Oriented R-CNN چند عامل کلیدی باعث افزایش کارایی محاسباتی (Efficiency) آن شده‌اند:

1. استفاده از **Oriented RPN سبک**
 - نقش: به جای استفاده از anchor های چرخیده متعدد (مانند RPN)، تنها از anchor های افقی با رگرسیون ۶ پارامتری استفاده می‌کند.
 - مزیت: کاهش شدید تعداد پارامترها و محاسبات – طبق مقاله، Oriented RPN تنها ۱/۳۰۰۰ پارامترهای RoI Transformer را دارد.

2. نمایش **Midpoint Offset**

- نقش: چرخش جعبه را بدون نیاز به زاویه یا مختصات پیچیده مدل می‌کند.
- مزیت: یادگیری پایدارتر و ساده‌تر، بدون بار محاسباتی اضافی از جمله محاسبه زوایا یا تبدیل هندسی.

3. تعداد مناسب **proposals**

- نقش: در مرحله اول، به جای ترکیب همه‌ی proposal ها، تنها ۱۰۰۰ پیشنهاد با نمره بالا استفاده می‌شوند.
- مزیت: کاهش زمان inference بدون افت قابل توجه در دقت.

- یک تحلیل انتقادی کوتاه از مقایسه Oriented R-CNN با سایر آشکارسازهای جهت دار دو مرحله ای که در مقاله نام برده شده اند ارائه دهید و نقاط ضعف و قوت را برجسته کنید .

در مقایسه با سایر مدل های دومرحله ای جهت دار مثل RoI Transformer+ ، Gliding Vertex ، Oriented R-CNN با سایر آشکارسازهای جهت دار مثل CAD-Net و RoI Transformer+ ، Gliding Vertex این مدل ارائه شده نقاط قوت و ضعف دارد:

- **نقاط قوت:**

садگی و سبکبودن: برخلاف RoI Transformer+ که سه مرحله مجزا و لایه های سنگین دارد، Oriented RPN تنها با افزودن دو آفست midpoint FC رگرسیون RPN سنتی کار می کند و $1/3000$ پارامتر RoI Transformer+ را دارد.

سرعت بالاتر: در آزمایش DOTA با عملکرد سریعی داشته .

دقت کافی : بدون نیاز به تنظیمات پیچیده، mAP برابر ۷۵.۸۷٪ را کسب کرده که بالاتر از اکثر روش های دومرحله ای مشابه است.

- **نقاط ضعف:**

حساسیت به مقیاس واحد آموزشی: برای رسیدن به 80.87% mAP، نیاز به آموزش و ارزیابی چند مقیاس دارد که هزینه محاسباتی و پیچیدگی پیاده سازی را بالا می برد.

عملکرد ناهمگن در برخی کلاس های کوچک : در کلاس هایی با اشیاء بسیار کوچک یا بسیار نامنظم (مثلًا Helicopter یا Roundabout)، گاهی کمی ضعیفتر از Gliding Vertex با R-101 عمل می کند.

در مجموع، Oriented R-CNN با تعادل منحصر به فردی بین سادگی، سرعت و دقت، یک baseline قوی برای آشکارسازی اشیاء مورب محسوب می شود، اما برای بیشینه کردن دقت در سناریوهای پسیار پیچیده یا با داده های چند مقیاس ممکن است نیاز به ترکیب با تکنیک های پیشرفته تر باشد.

۲-۵. پیاده سازی عملی (راه اندازی محیط و آماده سازی دیتاست)

دیتاست HRSC2016-MS

این دیتاست که شامل عکس های روتیت شده از کشتی ها بود را دانلود کردیم و سپس با استفاده از روش midpoint offset که در مقاله مطرح شده بود ، آن را لیبل زدیم.
درون هر عکس در دیتاست با استفاده از XML به صورت زیر لیبل گذاری شده بود :

```
<annotation>
    <contributors>Weiming Chen, Zizheng Ren, Bing Han, Zheng Yang, Ya
    <dataset>HRSC2016-MS</dataset>
    <source_dataset>HRSC2016</source_dataset>
    <label_level>L1</label_level>
    <filename>100000001</filename>
    <size>
        <width>1166</width>
        <height>753</height>
        <depth>3</depth>
    </size>
    <object>
        <type>bndbox</type>
        <name>ship</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>188</xmin>
            <ymin>244</ymin>
            <xmax>983</xmax>
            <ymax>510</ymax>
        </bndbox>
        <robndbox>
            <cx>582.8</cx>
            <cy>355.8349</cy>
            <w>189.1103</w>
            <h>778.2609</h>
            <angle>1.36</angle>
        </robndbox>
    </object>
</annotation>
```

شکل ۱۵ . مثالی از لیبل گذاری آبجکت ها با استفاده از فایل XML

لیبل زنی midpoint offset

حالا با استفاده از torch dataset اقدام به بارگذاری و لیبل زنی تصاویر کردیم . شرح مختصری از روش لیبل زنی به روش midpoint offset به صورت زیر است :

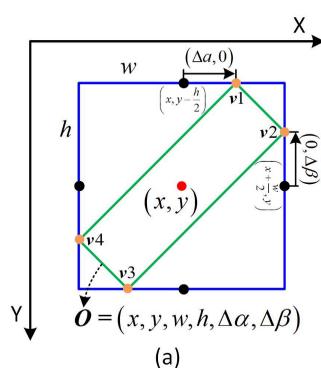
برای هر یک از آبجکت ها لیبل robndbox را پیدا کردیم که در واقع همان باندینگ باکس روتویت شده را نمایش میدهد.

```
robnd = obj.find('robndbox')
if robnd is None:
    continue
cx = float(robnd.find('cx').text)
cy = float(robnd.find('cy').text)
w = float(robnd.find('w').text)
h = float(robnd.find('h').text)
angle = float(robnd.find('angle').text)
```

شکل ۱۶. یافتن نقطه مرکز و طول و عرض عکس روتویت شده به همراه زاویه سپس اقدام به پیدا کردن چهار گوشه اصلی باکس روتویت شده کردیم با استفاده از :

```
cos_a = np.cos(angle)
sin_a = np.sin(angle)
dx, dy = w/2, h/2
corners = np.array([[-dx, -dy], [dx, -dy], [dx, dy], [-dx, dy]])
rot = np.array([[cos_a, -sin_a], [sin_a, cos_a]])
rot_corners = corners @ rot.T
trans = rot_corners + np.array([cx, cy])
```

شکل ۱۷. یافتن چهار نقطه کنترل باکس هر شی سپس با توجه به شکلی که در زیر از مقاله آمده است نیاز داشتیم که برای محاسبه ی آلفا و بتا از باکس اصلی که با محور های مختصات هم راست است نیز استفاده کنیم .



شکل ۱۸. روش محاسبه شش پارامتر اصلی هر باکس چرخدۀ

```

bnd = obj.find('bndbox')
if bnd is None:
    continue
w_h = float(bnd.find('xmax').text) - float(bnd.find('xmin').text)
h_h = float(bnd.find('ymax').text) - float(bnd.find('ymin').text)
top_mid = np.array([cx, cy - h_h/2])
right_mid = np.array([cx + w_h/2, cy])

```

شکل ۱۹ . یافتن باکس در برگیرنده کل شی، هم راستا با محور های مختصات عمودی و افقی

در واقع با استفاده از اختلاف باکس اصلی و چهار نقطه باکس چرخیده ، توانستیم آلفا و بتا را مشخص نماییم :

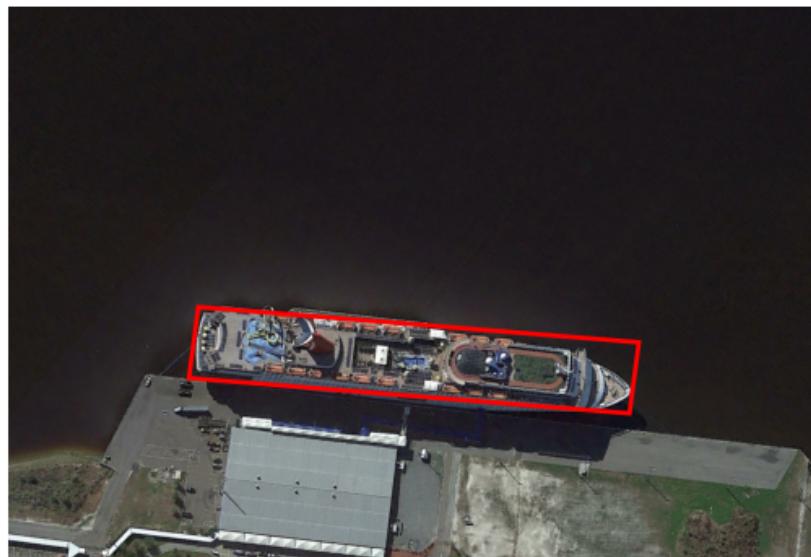
```

v1, v2 = trans[0], trans[1]
delta_alpha = v1[0] - top_mid[0]
delta_beta = v2[1] - right_mid[1]
midpoint_boxes.append([cx, cy, w_h, h_h, delta_alpha, delta_beta])

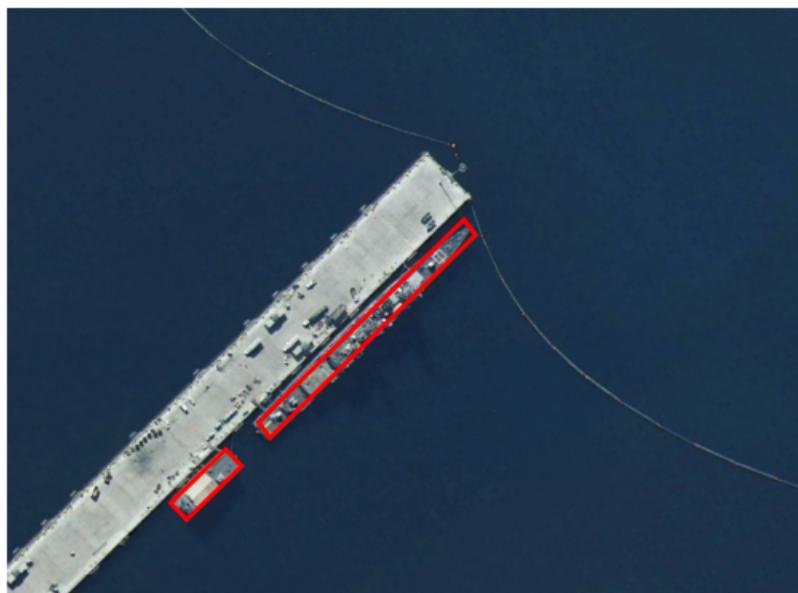
```

شکل ۲۰ . فرمول نهایی محاسبه آلفا و بتا

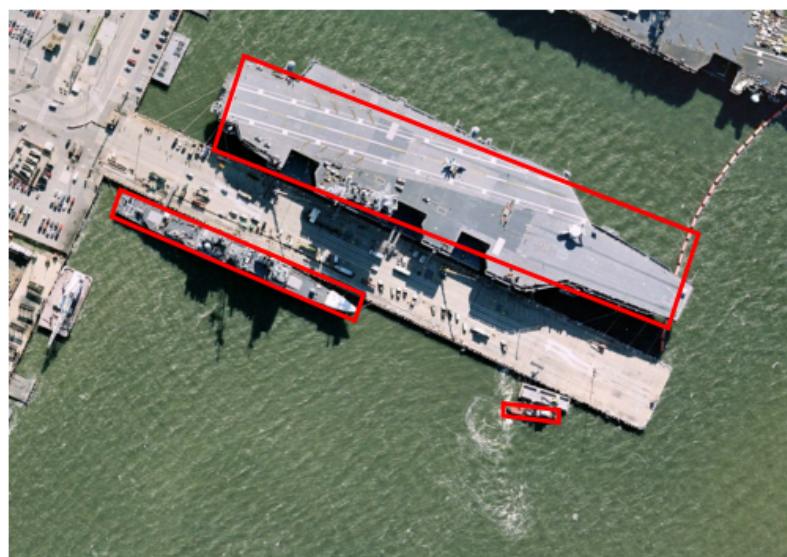
چند نمونه از لیبل زنی و باکس های آن
چند نمونه از لیبل زنی موفق به روش midpoint offset را در ادامه به تصویر میکشیم :



شکل ۲۱ . نمونه ای از کادر باندینگ باکس چرخیده شده



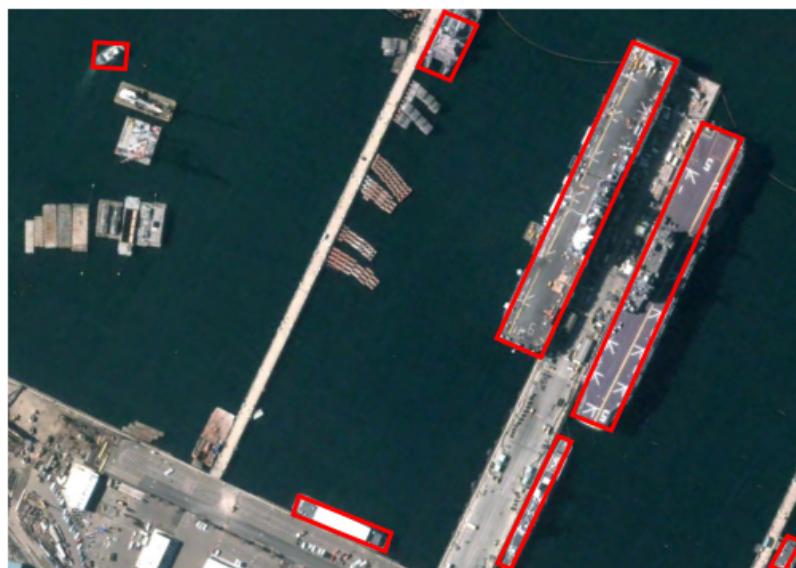
شکل ۲۲ . نمونه ای از کادر باندینگ باکس چرخیده شده



شکل ۲۳ . نمونه ای از کادر باندینگ باکس چرخیده شده



شکل ۲۴ . نمونه ای از کادر باندینگ باکس چرخیده شده



شکل ۲۵ . نمونه ای از کادر باندینگ باکس چرخیده شده

مدل ارائه شده در مقاله

مدلی که در مقاله ارائه شده بود را با استفاده از resnet-50-FPN بدون استفاده از پیاده سازی کردیم : MMDetection

```
class OrientedRCNN(nn.Module):
    def __init__(self, num_classes=2):
        super(OrientedRCNN, self).__init__()
        self.backbone = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

        if hasattr(self.backbone, 'backbone'):
            self.body = self.backbone.backbone.body
            self.fpn = self.backbone.backbone.fpn
        else:
            self.body = self.backbone.body
            self.fpn = self.backbone.fpn

        self.fpn_out_channels = 256
        self.rpn = OrientedRPN(in_channels=self.fpn_out_channels, num_anchors=3)
        self.head = OrientedRCNNHead(num_classes=num_classes, in_channels=self.fpn_out_channels)
        self.anchor_generator = AnchorGenerator()
```

شکل ۲۶. مدل ارائه شده ای Oriented RCNN

```
class OrientedRPN(nn.Module):
    def __init__(self, in_channels=256, num_anchors=3): # num_anchors is an argument here
        super().__init__()
        self.in_channels = in_channels
        self.num_anchors = num_anchors
        self.conv = nn.Conv2d(in_channels, in_channels, 3, padding=1)
        self.cls_logits = nn.Conv2d(in_channels, self.num_anchors * 2, 1) # Use self.num_anchors
        self.bbox_pred = nn.Conv2d(in_channels, self.num_anchors * 6, 1) # Use self.num_anchors
        for l in [self.conv, self.cls_logits, self.bbox_pred]:
            nn.init.normal_(l.weight, std=0.01)
            nn.init.constant_(l.bias, 0)
```

شکل ۲۷. قسمت Oriented RPN

```
class OrientedRCNNHead(nn.Module):
    def __init__(self, in_channels=256, num_classes=2):
        super().__init__()
        self.roi_align = RotatedRoIAlign(output_size=(7,7), spatial_scale=1/16)
        self.fc1 = nn.Linear(in_channels * 7 * 7, 1024)
        self.fc2 = nn.Linear(1024, 1024)
        self.cls_score = nn.Linear(1024, num_classes)
        self.bbox_refine = nn.Linear(1024, num_classes * 6)
```

شکل ۲۸. قسمت Oriented RCNN Head

با توجه به خواسته سوال ، مشخصات آموزش ما به شرح زیر اند :

- دیوایس استفاده شده : MPS (همان CUDA) است که بر روی GPU های مک اواس ران میشود .

- سایز Batch : هر یک از بچ های ما با اندازه یک بودند ، چون نمی توانستیم در قسمت یک بچ چند سایزی با اندازه های مختلف را به درستی خطاشوونو حساب کنیم .

تعداد Epoch ها : ما با توجه به اینکه روی Kaggle ران کردیم و محدودیت استفاده از GPU میخوردیم ، مجبور شدیم در سه راند که هر راند ۸ ایپاک بود ، مدل را آموزش دهیم. در دو راند اول مدل را سیو کردیم و در یک اکانت دیگر kaggle اقدام به آموزش مدل کردیم . و متاسفانه فقط دیتابی آموزش مربوط به راند سوم را در اختیار داریم . که نمودار های کشیده شده ، از این راند هستند .

هر ایپاک ما دقیقا 1 ساعت طول میکشد !!!

- نرخ یادگیری (Learning Rate) : نرخی که ما استفاده کردیم 0.002 بود . (که نرخ متداولی در پروژه های مان است) (در واقع ما ، در همه پروژه ها از این نرخ استفاده کردیم)

- وزن Weight Decay : به اندازه $1e-4$ ، که در SGD کاربرد دارد .

- وزن momentum که در backpropagation در SGD به اندازه 0.9 است کردیم .

```
Epoch 1/8: 100%|██████████| 610/610 [56:57<00:00,  5.60s/it, loss=0.186]
[Epoch 1] Avg Loss: 0.4249 | RPN Acc: nan
Epoch 2/8: 100%|██████████| 610/610 [56:53<00:00,  5.60s/it, loss=0.257]
[Epoch 2] Avg Loss: 0.3804 | RPN Acc: nan
Epoch 3/8: 100%|██████████| 610/610 [56:41<00:00,  5.58s/it, loss=0.344]
[Epoch 3] Avg Loss: 0.3612 | RPN Acc: nan
Epoch 4/8: 100%|██████████| 610/610 [56:40<00:00,  5.58s/it, loss=0.559]
[Epoch 4] Avg Loss: 0.3483 | RPN Acc: nan
Epoch 5/8: 100%|██████████| 610/610 [56:45<00:00,  5.58s/it, loss=0.231]
[Epoch 5] Avg Loss: 0.3295 | RPN Acc: nan
Epoch 6/8: 100%|██████████| 610/610 [57:18<00:00,  5.64s/it, loss=0.136]
[Epoch 6] Avg Loss: 0.2789 | RPN Acc: nan
```

شکل ۲۹. اطلاعات در حال آموزش

همان طور که در شکل بالا دیده میشود ، مدل در هر ایپاک در حال بهتر شدن و به سمت جنرالیزیشن است .



شکل ۳۰. اطلاعات تابع هزینه در حین آموزش راند سوم ۸ ایپاک (در واقع ایپاک ۲۴ ام در کل)

همان طور که توضیح دادم :

ما با توجه به اینکه روی Kaggle ران کردیم و محدودیت استفاده از GPU میخوردیم ، مجبور شدیم در سه راند که هر راند ۸ ایپاک بود ، مدل را آموزش دهیم. در دو راند اول مدل را سیو کردیم و در یک اکانت دیگر kaggle اقدام به آموزش مدل کردیم . و متاسفانه فقط دیتابی آموزش مربوط به راند سوم را در اختیار داریم . که نمودار های کشیده شده ، از این راند هستند.

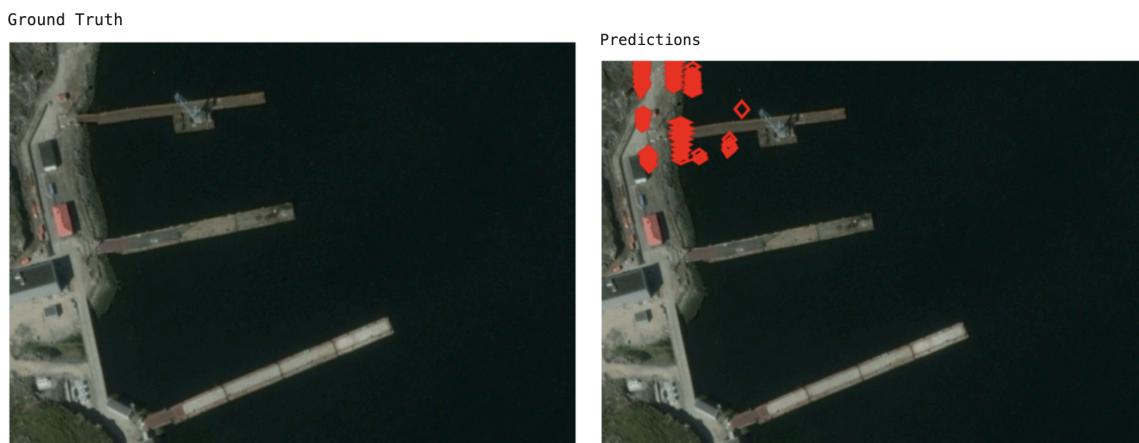
هر ایپاک ما دقیقا یک ساعت طول میکشد.

- روند نزولی یکنواخت: از ابتدا تا انتهای، تابع هزینه کاهش پیوسته‌ای دارد که نشانه‌ای از یادگیری موفق مدل است.
- عدم وجود نوسانات شدید: مسیر کاهش Loss صاف و بدون جهش یا افزایش ناگهانی است، که نشان‌دهنده پایداری فرآیند آموزش و نداشتن اختلال در گرادیان یا داده است.
- همگرایی: کاهش مداوم مقدار loss نشانه‌ای از همگرایی مناسب مدل است، بهویژه در اواخر نمودار که شبکه کاهش ملایمتر می‌شود، یعنی مدل به آرامی به یک نقطه بهینه نزدیک می‌شود.
- روند کاهش یکنواخت و نبود نوسان‌های زیاد، نشان‌دهنده تنظیم مناسب پارامترهای آموزش مانند نرخ یادگیری (learning rate) و الگوریتم بهینه‌سازی است.

- مدل دچار افت عملکرد ناگهانی یا نوسانات نشده که نشانه خوبی از پایداری الگوریتم آموزش است.

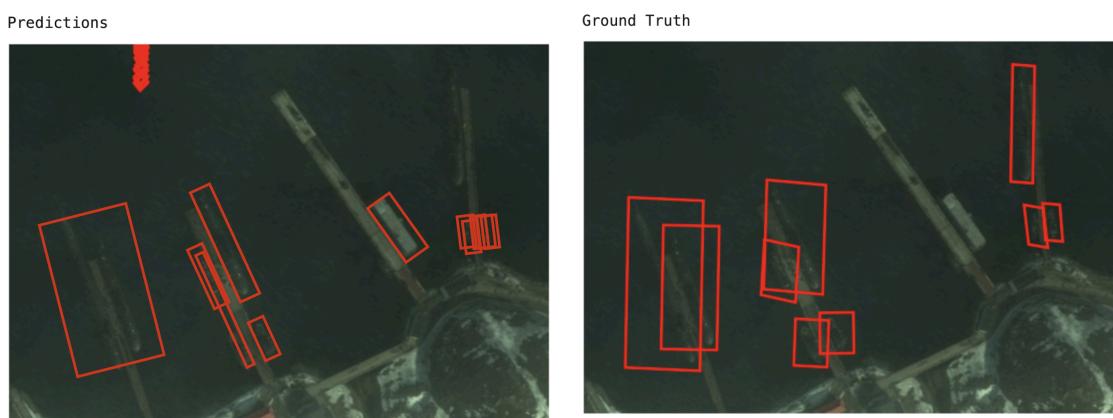
ارزیابی مدل و تحلیل نتایج

چند تصویر از خروجی های مدل در داده تست به صورت زیر اند :



شکل ۳۱ .نمونه اول از خروجی مدل

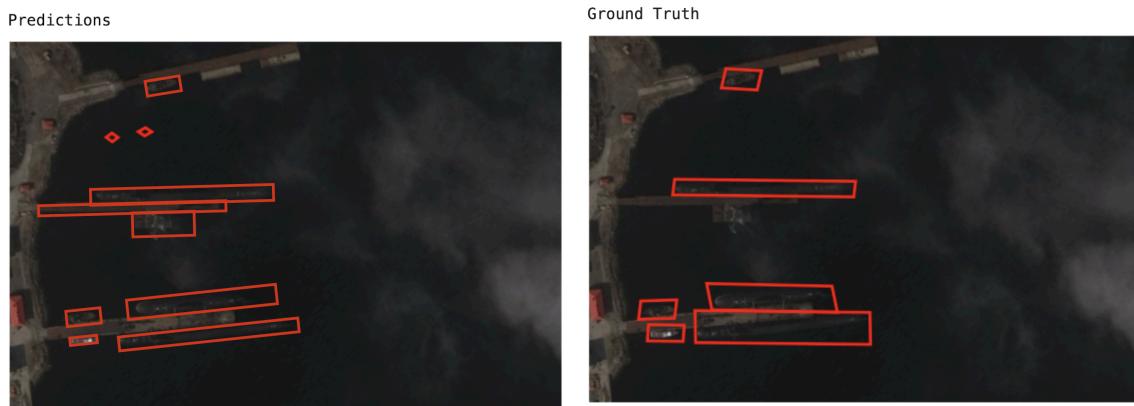
در این نمونه از داده تست ، هیچ کشتی ای در تصویر مشاهده نمیشود اما مدل ما به اشتباہ چندین bounding box پیشنهاد میدهد . که احتمالا مشکل از این است که در هنگام آموزش داده ای که شامل هیچ کشتی ای نباشد را کم دیده است که توانایی جنرالیسیزین آن را داشته باشد.



شکل ۳۲ .نمونه دوم از خروجی مدل

در این نمونه داده حتی مشاهده میشود که مدل ما خروجی بهتری از خروجی GT داده است چون که چرخش های باکس ها را بهتر تشخیص داده است ، اما در قسمت بالایی تصویر

چندیden باکس اشتباه تولید کرده است ، همچنین در سمت راست چند کشتی کوچک را به تعداد بالا شناسایی کرده است که در صورتی که مدل ما فقط دو تا کشتی کج داشته . البته که چندین کشتی را هم ندیده است .



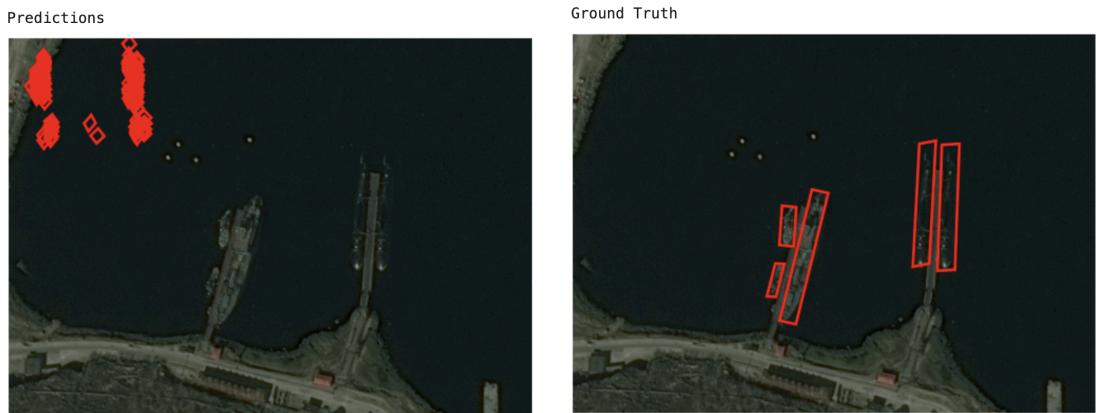
شکل ۳۳ . نمونه سوم از خروجی مدل

در نمونه بالا نیز مدل با دقت بالایی خوب عمل کرده است اما چندین باکس اضافه به عنوان کشتی کشیده است . که به نسبت GT اضافه تر بودند .



شکل ۳۴ . نمونه چهارم از خروجی مدل

در این نمونه مدل کاملا اشتباه رفتار کرده است علت آن هم میتواند دقت پایین مدل باشد که احتمالا با train بیشتر بهتر میگردد .



شکل ۳۵ . نمونه پنجم از خروجی مدل

در این نمونه هم ، مدل کاملاً اشتباه رفتار کرده است علت آن هم میتواند دقت پایین مدل باشد که احتمالاً با train بیشتر بهتر میگردد .

حالا با استفاده از قطعه کد زیر اقدام به ارزیابی مدل میکنیم :

```
def evaluate_model(model, dataloader, device):
    model.train()
    total_loss = 0.0
    with torch.no_grad():
        for images, targets in tqdm(dataloader, desc="Evaluating"):
            images = images.to(device)
            targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

            losses = model(images, targets)
            loss = sum(losses.values())
            total_loss += loss.item()

    avg_loss = total_loss / len(dataloader)
    print(f"Average Evaluation Loss: {avg_loss:.4f}")
```

شکل ۳۶ .تابع ارزیابی برای مدل بر روی دادگان تست

حالا به سراغ ارزیابی مدل بر روی دادگان تست میرویم که به صورت زیر پس از ۴۵ دقیقه است :

```
evaluate_model(model, test_loader, device)
```

```
Evaluating: 100%|██████████| 610/610 [44:20<00:00, 4.36s/it]
Average Evaluation Loss: 0.6330
```

شکل ۳۷. دقت مدل بر روی دادگان تست

تحلیل مقایسه ای و پیشنهاد بهبود

- عملکرد مدل خود را با معیار های ارائه شده در مقاله مقایسه کنید . نتایج خود را به طور واضح بیان کنید و تفاوت های احتمالی با مقاله را بررسی کنید . به دلیل احتمالی این تفاوت ها اشاره و پیشنهاد های مشخص و عملی برای بهبود پیاده سازی خود ارائه بدهید .

مقایسه عملکرد:

دقت مدل ما (accuracy) برابر 63.30 % شده است. برای مقایسه مؤثر، باید به دقت مدل اصلی مقاله در همان مجموعه داده نگاه کنیم.

بهطور معمول در مقاله ها، معیارهایی مثل IoU، mAP (mean Average Precision)، استفاده می شود، نه فقط Accuracy، بهخصوص برای مدل های تشخیص شیء.

اگر مقاله اصلی از mAP استفاده کرده باشد و مقدار آن بالاتر (مثلاً 75٪ یا بیشتر) باشد، تفاوت عملکرد مشخص خواهد بود.

حتی اگر مقاله دقت مشابهی گزارش کرده باشد، ممکن است با استفاده از داده های بیشتر، تکنیک های loss function regularization، augmentation باشد.

تحلیل تفاوت ها با مقاله:

تفاوت در داده‌ها:

نسخه‌ای از دیتاست که از کشتی استفاده کردیم با نسخه مقاله متفاوت است (در تعداد تصاویر، کیفیت، یا توازن بین کلاس‌ها و نوع عکس‌های کشتی - اما در مقاله هلیکوپتر و هواپیما هم بود).

:loss function ساده:

بخش `compute_rcnn_loss` کامل پیاده‌سازی نشده (فقط مقدار صفر برمی‌گرداند)، و این در آموزش `head` نقش مهمی دارد.

:data augmentation کمبود

مقالات‌ها معمولاً از تکنیک‌های غنی‌سازی تصویر مانند `color`, `rotation`, `scale jittering` و `distortion` برای بهبود عملکرد استفاده می‌کنند.

:iteration / epoch کم بودن

قطععاً مدل به اندازه کافی آموزش ندیده است ($8 * 610$ در مقابله با $20k$ در مقاله).

پیشنهادات عملی برای بهبود :

تکمیل loss head:

حتماً تابع `compute_rcnn_loss` را دقیق تر کنیم تا `head` نیز آموزش ببیند.

افزایش و تنوع دادهها:

از `data augmentation` مخصوص اشیاء متمایل استفاده کنیم : چرخش، مقیاس‌بندی تصادفی، `flipping` مورب.

جون در این دیتابست ، جرخاندن عکس ها خیلی خوب دیتا میسازد

افزایش تعدد آموزش:

تعداد epoch یا `iteration` را افزایش دهیم و از `learning rate schedule` استفاده کنیم.

تحلیل دقیق‌تر با mAP:

از معیارهای ارزیابی دقیق‌تر مانند $mAP@IoU=0.5:0.95$ استفاده کنیم تا جزئیات بهتری از عملکرد به دست آوریم.