

به نام خدا



دانشگاه تهران



دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

آرمنین قاسمی	نام و نام خانوادگی	پرسش ۱
۸۱۰۱۰۰۱۹۸	شماره دانشجویی	
امیرحسین ثمودی	نام و نام خانوادگی	پرسش ۲
۸۱۰۱۰۰۱۰۸	شماره دانشجویی	
۱۴۰۳.۱۲.۲۷	مهلت ارسال پاسخ	

فهرست

پرسش ۱. سگمنتیشن تصاویر شهری

۴	۱-۱. توصیف مدل ارائه شده
۶	۲-۱. آماده سازی مجموعه داده
۷	۳-۱. بهینه ساز، متریک ها و تابع هزینه
۸	۴-۱. پیاده سازی مدل
۱۰	۵-۱. آموزش مدل
۱۵	۶-۱. ارزیابی مدل

پرسش ۲ – Oriented R-CNN برای تشخیص اشیا

۲۰	بخش اول: سوالات نظری
۲۰	۲۰ مقدمه
۲۰	درک مفهومی
۲۱	۲۱ اجزای مدل
۲۴	۲۴ Rotated RoI Align
۲۶	۲۶ عملکرد و کارایی
۲۸	بخش دوم: پیاده سازی عملی
۲۸	۲۸ راه اندازی محیط و آماده سازی مجموعه داده
۳۴	۳۴ آموزش مدل Oriented R-CNN

شکل‌ها

۵ شکل ۱: معماری Fast-SCNN (تصویر موجود در مقاله)
۶ شکل ۲: یک نمونه داده همراه با ماسک
۷ شکل ۳: نمایش نحوه data augmentation
۱۱ شکل ۴ نمودار یادگیری با بهینه ساز adam
۱۱ شکل ۵: نمودار یادگیری با بهینه ساز sgd
۱۶ شکل ۶: تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز adam) تصاویر ۱ تا ۵
۱۷ شکل ۷: تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز adam) تصاویر ۶ تا ۱۰
۱۸ شکل ۸: تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز sgd) تصاویر ۱ تا ۵
۱۹ شکل ۹: تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز sgd) تصاویر ۶ تا ۱۰
۲۲ شکل ۱۰: معماری Oriented RPN
۲۵ شکل ۱۱: مراحل Rotated RoIAlign
۲۶ شکل ۱۲: استفاده از Horizontal ROI
۲۸ شکل ۱۳: نمونه‌ای از یک فایل Annotation
۲۹ شکل ۱۴: بارگذاری داده‌های یک نمونه تصویر
۳۰ شکل ۱۵: مختصات رئوس باکس چرخش یافته
۳۱ شکل ۱۶: تعیین مقادیر آفست
۳۱ شکل ۱۷: نحوه چرخش و بدست آوردن مختصات رئوس و مقادیر آفست
۳۲ شکل ۱۸: نحوه تبدیل نمایش Midpoint offset برای رسم باکس‌ها
۳۲ شکل ۱۹: بارگذاری نمونه اول
۳۳ شکل ۲۰: بارگذاری نمونه دوم
۳۳ شکل ۲۱: بارگذاری نمونه سوم

جدول‌ها

۵Fast SCNN و U-Net مقایسه : جدول ۱

۹bottleneck residual block به مربوط مقاله : جدول ۲

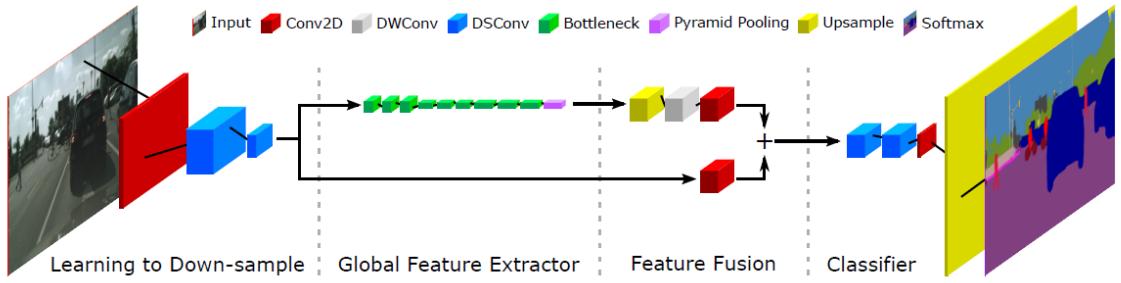
پرسش ۱. سگمنتیشن تصاویر شهری

۱-۱. توصیف مدل ارائه شده

مدلی که در مقاله مربوطه معرفی شده، یک مدل سبک و سریع (real time) برای سگمنتیشن معنایی تصاویر شهری است. این مدل عملکرد خوبی داشته به طوری که روی داده های Cityscapes به IoU ۶۸٪ رسیده و سرعت پردازش آن ۱۲۳.۵ فریم بر ثانیه است. همچنین این مدل نسبت به مدل های مشابه خود تعداد پارامتر های بسیار کمتری دارد.

تقریبا تمام مدل های Semantic Segmentation می خواهند معماری ارائه دهنده بتواند بر اساس ویژگی های اصلی object ها (که در لایه های عمیق تر بدست می آید) object را تشخیص داده و شناسایی کند و بر اساس ویژگی های سطح پایین تر مانند لبه ها و گوشه ها (که در لایه ها اولیه و کم عمق تر بدست می آید) مرز های بین تصاویر را تشخیص دهنده. یعنی در واقع مدل باید بتواند هم ویژگی های لایه های عمیق تر و هم ویژگی های لایه های کم عمق تر را داشته باشد که بتواند بر اساس آنها نتیجه خوبی ارائه دهد. برای این کار مدل های encoder-decoder پیشنهاد شدند که با استفاده از skip connection های متعدد می توانند ویژگی های استخراج شده از لایه ها کم غمق تر را در اختیار لایه های عمیق تر قرار دهند. تا بتوان از خروجی های لایه های اولیه نیز استفاده کرد. این مدل ها سرعت پایینی دارند و از حافظه بسیار زیادی استفاده میکنند. همچنین توان محاسباتی بسیار بالایی نیز دارند و پیاده سازی آنها در سیستم ها بدون GPU تقریبا غیر ممکن است. با پیشرفت تکنولوژی خودرو های خودران احساس نیاز به سیستم های real time برای semantic segmentation ایجاد شد. برای این که مدل ها سریع تر و سبک تر باشند، مدل های چند شاخه ای (و خصوصاً دو شاخه ای) پیشنهاد شدند. معماری این مدل ها اینگونه است که دو شاخه یکی با تعداد لایه ها و عمق بیشتر (برای ویژگی های اصلی و کلی) و یکی با تعداد لایه ها و عمق کمتر (برای بدست آوردن مرز ها)، داده ها را پردازش میکنند و نهایتاً خروجی شاخه ها با یکدیگر ادغام میشوند و خروجی نهایی از روی آن بدست می آید. بدین ترتیب هم سرعت پردازش و هم استفاده از حافظه و نیاز به توان محاسباتی کاهش می یابد.

مدل Fast – SCNN از مزایای هر کدام از معماری های قبلی استفاده کرده، و مدلی ارائه داده که هم سرعت بالا و هم دقت مناسبی دارد. معماری این مدل در شکل ۱ قابل مشاهده است.



شکل ۱: معماری Fast-SCNN (تصویر موجود در مقاله)

می توانیم بگوییم که این مدل به طور کلی از دو شاخه spatial (شاخه کم عمق تر) و global (شاخه عمیق تر) تشکیل شده است. شاخه اول ویژگی های هندسی و شاخه دوم ویژگی های کلی را در بر دارد.

ابتکار این مدل مازول learning to down sample است. این مازول بین هر دو شاخه مشترک است و ویژگی های سطح پایین مانند لبه ها، بافت ها و گوشه ها را استخراج میکند. همچنین رزولوشن را برای استفاده در مازول بعدی کاهش میدهد. مازول global feature extractor مربوط به استخراج ویژگی های کلی است که همان شاخه عمیق تر ما است. این شاخه از تعدادی بلاک bottleneck residual و یک feature fusion Pyramid Pooling تشکیل شده است که مسئول استخراج ویژگی های کلی هستند. مازول feature fusion ویژگی های سطح پایین تر و ویژگی های کلی تر را با هم ادغام میکند و نهایتاً classifier نیز خروجی نهایی که تصویری به همان اندازه تصویر ورودی است را به ما میدهد.

می توانیم U-Net و Fast-SCNN را به این شکل با هم مقایسه کنیم:

جدول ۱ : مقایسه Fast SCNN و U-Net

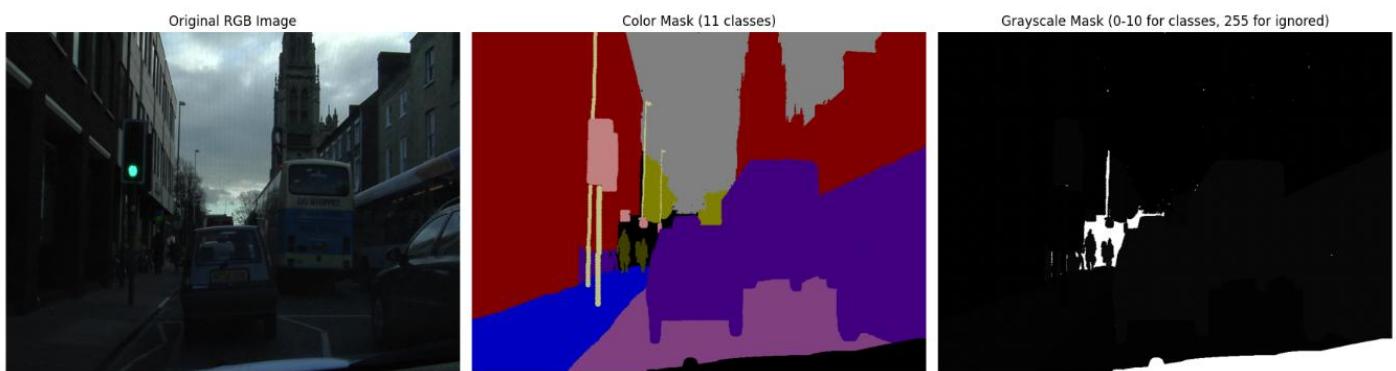
U-Net	Fast-SCNN	مدل ویژگی
Encoder-Decoder	Two-Branch	نوع معماری
دقت بالا در segmentation پزشکی و عمومی	Real-time segmentation (عموماً برای تصاویر شهری و سیستم های خودران)	هدف
بهینه برای دقت	بهینه برای سرعت	طراحی
skip connection encoder ↔ decoder	فقط یک skip connection اولیه	Skip connections

۱-۲. آماده سازی مجموعه داده

مجموعه داده ها را دریافت میکنیم و تعداد آنها را گزارش میکنیم:

Training samples: 367
Validation samples: 101
Test samples: 233

یک نمونه داده را هم همراه با ماسک های سیاه سفید و رنگی نمایش میدهیم:



شکل ۲ : یک نمونه داده همراه با ماسک

در واقع ماسک های رنگی صرفا برای بصری سازی هستند و تمام مراحل ترین و تست مدل با ماسک های سیاه سفید انجام میشود. در ماسک های سیاه سفید ما هر کلاس را با یک رنگ و در یک کانال نشان میدهیم مثلاً رنگ ۰ برای ماشین ها رنگ ۱ برای درخت و... . همچنین با رنگ سفید (۲۵۵) مواردی که مهم نیستند و باید ایگنور شوند را نشان میدهیم.

با توجه به اینکه تعداد این داده ها زیاد نیست data augmentation را روی این داده ها انجام میدهیم.

تغییرات زیر را روی داده ها اعمال می کنیم:

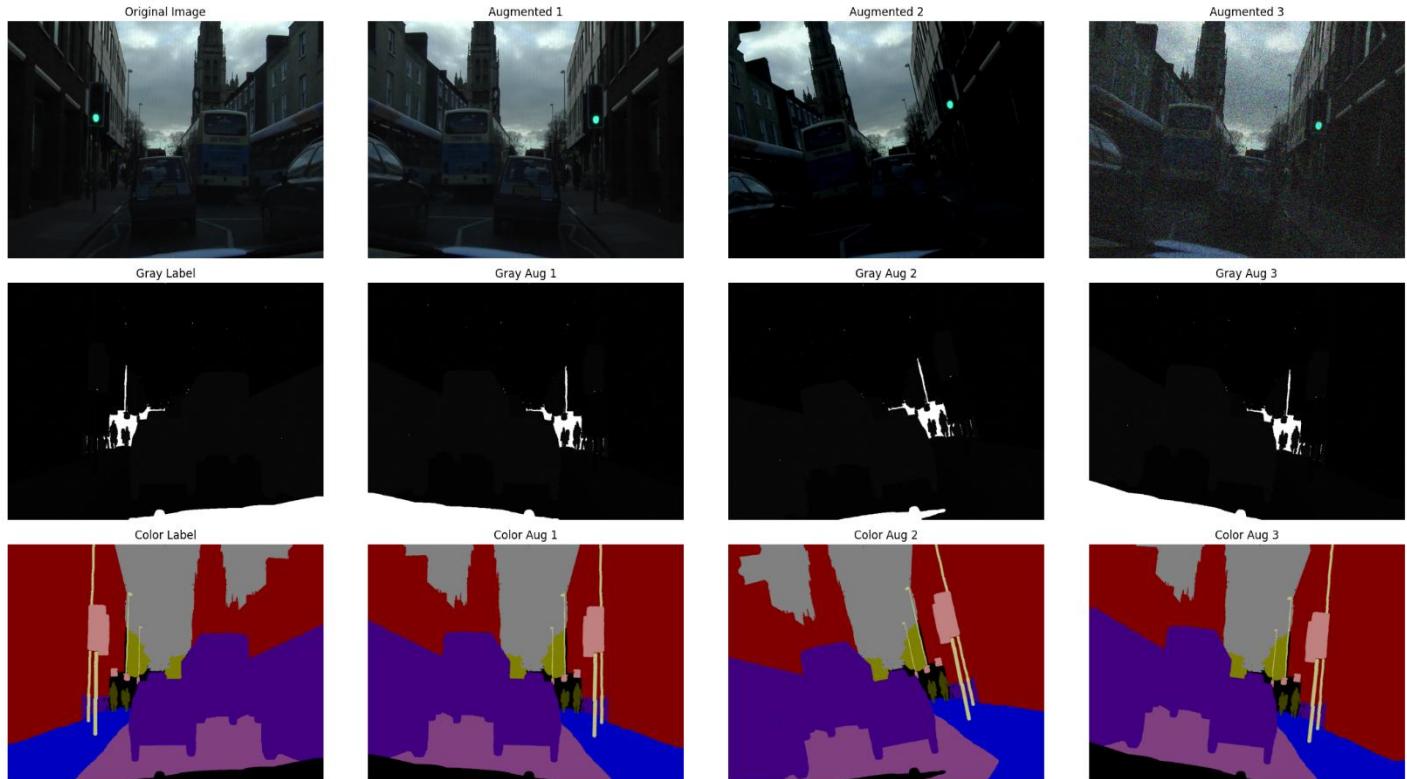
- تغییر شدت نور
- ایجاد کمی نویز روی تصاویر
- قرینه کردن افقی
- چرخاندن تصویر به مقدار کم

با استفاده از این تغییرات داده ها را تا ۴ برابر افزایش میدهیم. نکات مهمی که باید در اعمال این تغییرات درنظر بگیریم این موارد هستند:

- تغییرات هندسی مانند فلیپ کردن و چرخاندن باید روی ماسک ها هم اعمال شوند ولی تغییراتی مانند تغییر نور یا نویز صرفا روی تصاویر اصلی اعمال میشوند.

- در چرخاندن تصاویر نباید حاشیه سیاه برای تصاویر درست کنیم و در واقع باید وسط تصویر را برش دهیم و مجددا resize کنیم.

یک نمونه داده هماره با ماسک ها و نمونه های augment شده آن در زیر آورده شده:



شکل ۳ : نمایش نحوه data augmentation

با توجه به شکل نمونه Aug1 به صورت افقی قرینه شده، نمونه Aug2 هم قرینه شده، هم چرخیده و هم تاریک تر شده، نمونه Aug3 نیز چرخیده و نویز کمی روی آن سوار شده.

نهایتا فایل های تکست مربوطه را مانند قبل می سازیم که داده های تست و ترین و اعتبارسنجی از یکدیگر جدا شوند.

۱-۳. بهینه ساز، متریک ها وتابع هزینه

IoU در واقع نسبت اشتراک به اجتماع ماسک واقعی و پیشビینی شده است. در واقع می توان گفت نسبت ناحیه ای که درست پیشビینی شده، نسبت به ناحیه ای که پیشビینی شده یا جزو کلاس مورد نظر بوده. Dice Coefficient نیز نسبت دو برابر اشتراک ماسک واقعی نسبت به حاصل جمع ناحیه های پیشビینی شده و ناحیه واقعی است. این دو معیار به طور کلی سختگیر تر از accuracy هستند. کد پیاده سازی این دو معیار در نوت بوک آپلود شده موجود است.

ما در پیاده سازی ازتابع هزینه crossentropy و بهینه ساز learning rate adam با متغیر استفاده کردیم و batch size را نیز ۳۲ قرار دادیم. البته یک نمونه از مدل نیز با sgd آموزش داده شد ولی چون نتایج adam بهتر بود این مدل را به عنوان معیار درنظر میگیریم.

۴-۱. پیاده سازی مدل

مدل مطابق مقاله پیاده سازی شد و آن گرفته شد. مطابق این گزارش، مدل مورد نظر ۴۵۷۴۶۷ پارامتر آموزش پذیر دارد.

Layer (type:depth-idx)	Output Shape	Param #
<hr/>		
FastSCNN	[1, 11, 960, 720]	--
LearningToDownsample: 1-1	[1, 64, 120, 90]	--
Sequential: 2-1	[1, 64, 120, 90]	--
Conv2d: 3-1	[1, 32, 480, 360]	864
BatchNorm2d: 3-2	[1, 32, 480, 360]	64
ReLU: 3-3	[1, 32, 480, 360]	--
DSConv: 3-4	[1, 48, 240, 180]	1,984
DSConv: 3-5	[1, 64, 120, 90]	3,728
GlobalFeatureExtractor: 1-2	[1, 128, 30, 23]	--
Sequential: 2-2	[1, 128, 30, 23]	--
InvertedResidual: 3-6	[1, 64, 60, 45]	54,272
InvertedResidual: 3-7	[1, 96, 30, 23]	66,624
InvertedResidual: 3-8	[1, 128, 30, 23]	136,768
PyramidPoolingModule: 2-3	[1, 128, 30, 23]	--
ModuleList: 3-9	--	66,560
Conv2d: 3-10	[1, 128, 30, 23]	82,048
FeatureFusionModule: 1-3	[1, 128, 120, 90]	--
Sequential: 2-4	[1, 128, 30, 23]	--
Conv2d: 3-11	[1, 128, 30, 23]	16,384
BatchNorm2d: 3-12	[1, 128, 30, 23]	256
Sequential: 2-5	[1, 128, 120, 90]	--
Conv2d: 3-13	[1, 128, 120, 90]	8,192
BatchNorm2d: 3-14	[1, 128, 120, 90]	256
ReLU: 2-6	[1, 128, 120, 90]	--
Classifier: 1-4	[1, 11, 120, 90]	--
Sequential: 2-7	[1, 11, 120, 90]	--
Conv2d: 3-15	[1, 128, 120, 90]	1,152
BatchNorm2d: 3-16	[1, 128, 120, 90]	256
ReLU: 3-17	[1, 128, 120, 90]	--
Conv2d: 3-18	[1, 128, 120, 90]	16,384
BatchNorm2d: 3-19	[1, 128, 120, 90]	256
ReLU: 3-20	[1, 128, 120, 90]	--
Conv2d: 3-21	[1, 11, 120, 90]	1,419
<hr/>		
Total params: 457,467		
Trainable params: 457,467		
Non-trainable params: 0		
Total mult-adds (Units.GIGABYTES): 1.16		
<hr/>		
Input size (MB): 8.29		
Forward/backward pass size (MB): 354.38		
Params size (MB): 1.83		
Estimated Total Size (MB): 364.50		
<hr/>		

وظایف هر یک از ماظول ها در بخش توصیف مدل ارائه شده بیان شد. در این بخش معماری درون هر یک از این ماظول هارو بررسی میکنیم:

- : این ماظول از سه لایه convolution تشکیل شده که یک لایه به LearningtoDownsample صورت Conv2D یا همان لایه convolution استندارد است و دو لایه دیگر به صورت DSConv

یا همان DSConv depthwise separable convolutional layer است. مازول های DSConv برای افزایش سرعت و کاهش استفاده از مموری به کار میروند. این مازول ها از دو لایه کانولوشنی Depthwise و Pointwise در کنار هم استفاده می‌کنند. فرض کنید می خواهیم روی ورودی با 64×64 کanal یک فیلتر 3×3 در 3×3 اعمال کنیم و خروجی ای با 128×128 کanal بسازیم. اگر یک فیلتر کانولوشنی معمولی اعمال کنیم به $64 \times 128 \times 3 \times 3 = 73728$ پارامتر نیاز خواهیم داشت. ولی در روش DSConv ابتدا یک فیلتر سه در سه را روی تک تک کanal ها اعمال میکنیم. یعنی $64 \times 3 \times 3$ در 3×3 . نهایتاً روی خروجی یک فیلتر یک در یک 64×64 تایی به 128×128 اعمال میکنیم. در این حالت مجموع پارامتر های ما 8768 پارامتر خواهد بود که تقریباً یک دهم مورد قبلی است. علت اینکه در لایه اول از DSConv استفاده نمی کنیم این است که تعداد کanal های ورودی فقط 3×3 کanal است و استفاده از این تکنیک تاثیر خاصی روی تعداد پارامتر ها ندارد. سایز تمامی فیلتر های اعمال شده 3×3 و آنها 2 است که همراه با $ReLU$ و batch normalization stride 2 روی داده های ورودی اعمال میشوند. رزولوشن نهایی این مازول یک هشتم رزولوشن ورودی است.

- GlobalFeatureExtractor : در این مازول از سه bottleneck residual block استفاده شده است که هم زمان با افزایش کanal ها اندازه تصویر را کاهش میدهند (مثل یک قیف عمل میکنند) در این مازول ها نیز از DSConv استفاده میشود که محاسبات سبک تر باشند. جدول مربوط به معماری این مازول ها در زیر آورده شده:

جدول ۲ : جدول ۲ مقاله مربوط به bottleneck residual block

Input	Operator	Output
$h \times w \times c$	Conv2D $1/1, f$	$h \times w \times tc$
$h \times w \times tc$	DWConv $3/s, f$	$\frac{h}{s} \times \frac{w}{s} \times tc$
$\frac{h}{s} \times \frac{w}{s} \times tc$	Conv2D $1/1, -$	$\frac{h}{s} \times \frac{w}{s} \times c'$

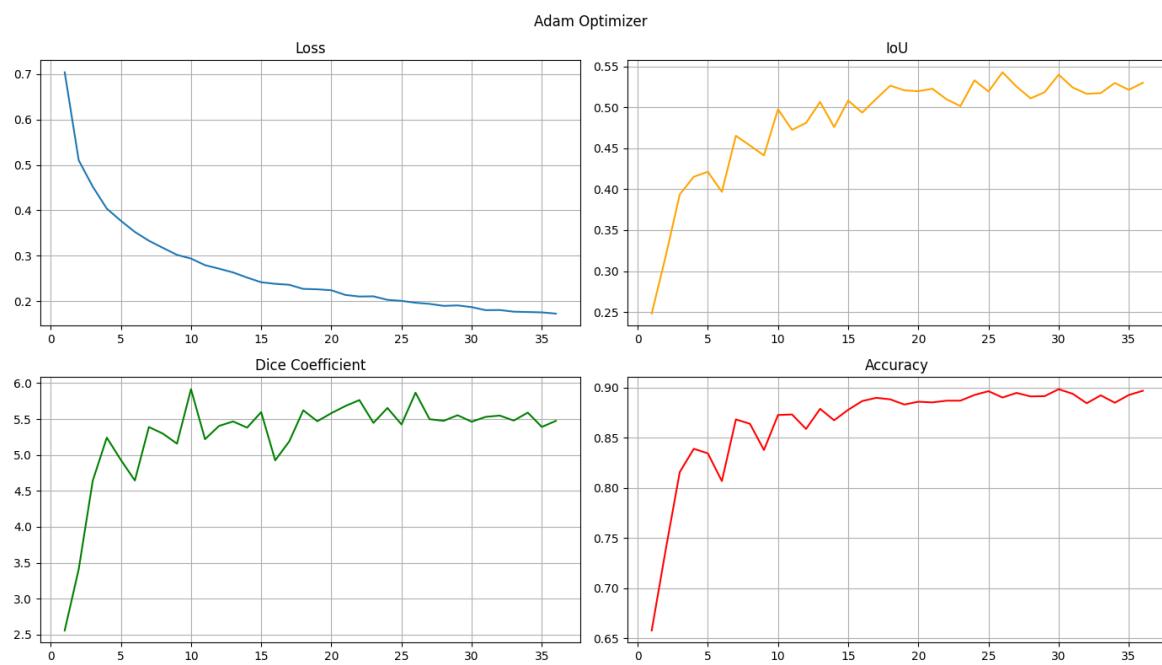
با توجه به این معماری ابتدا کanal های ورودی را افزایش میدهیم، سپس با استفاده از DSConv (منظور از DSConv همان دو سطر آخر در جدول است) کانولوشن را روی کanal های جداگانه اعمال می کنیم و اگر خواستیم تعداد کanal ها را کاهش میدهیم. در واقع با depthwise conv تعداد کanal ها را افزایش داده و با pointwise conv مجدداً تعداد کanal ها را کاهش میدهیم و اگر سایز ورودی و خروجی برابر باشد، ورودی را با خروجی جمع میکنیم (residual connection).

در انتهای این مژول از Pyramid Pooling نیز استفاده میکنیم. با توجه به اینکه مقاله در باره این ساختار توضیح خاصی نداده باد معماری این بخش را اینگونه اعمال می کنیم:

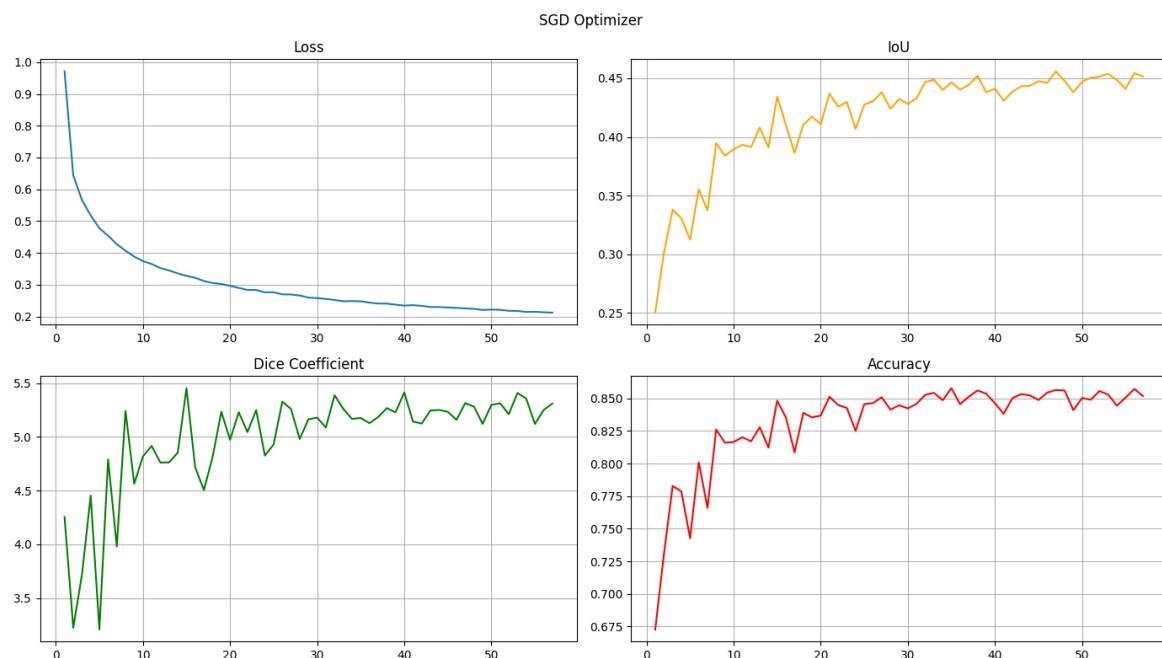
- از feature map خروجی چند نسخه با رزولوشن پایین تر می سازیم. (با استفاده از با Average Pooling $(1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6)$)
 - هر نسخه را با یک کانولوشن ۱ در ۱ ، Relu و batch normalization پردازش میکنیم.
 - هر نسخه را به اندازه ورودی resize میکنیم.
 - همه نسخه ها را با نسخه اصلی concat میکنیم.
 - از یک Bottleneck (1×1 Conv) برای ترکیب نهایی استفاده می کنیم.
- FeatureFusionModule : این مژول صرفا به ساده ترین شکل ممکن خروجی های LearningtoDownsample و GlobalFeatureExtractor را resize میکند که یک اندازه شوند و بتوان آنها را جمع کرد و سپس آنها را با هم جمع میکند.
- Classifier : Classifier - depthwise separable convolution از دو لایه pointwise convolution استفاده می کنیم و نهایتا یک تصویر با همان سایز تصویر ورودی میسازیم که مقدار هر پیکسل، معرف کلاس آن پیکسل است.

۱-۵. آموزش مدل

مدل را با تعداد 100 ایپاک آموزش میدهیم. البته برای مدل مکانیزم Early stopping را در نظر میگیریم که اگر پس از ۱۰ ایپاک عملکرد مدل روی داده های validation بهتر نشد، یادگیری متوقف شود. مدل را با دو optimizer adam و sgd به ترتیب با learning rate های 0.01 و 0.005 ترین میکنیم. البته با استفاده از رابطه $lr = init_lr * (1 - iter / total_iters)^{** 0.9}$ در هر گام learning rate را آپدیت می کنیم. batch size را نیز برابر با ۳۲ قرار میدهیم. نمودار ها و لاغ های مربوط به آموزش مدل در ادامه آورده شده است.



شکل ۴ نمودار یادگیری با بهینه ساز adam



شکل ۵ : نمودار یادگیری با بهینه ساز sgd

با توجه به این نمودار ها مدل با بهینه ساز adam سریع تر ترین شده و به دقت بالاتری رسیده. در ادامه لگ های مربوط به آموزش مدل را بررسی میکنیم و مقدار دقیق متريک های موردنظر را گزارش میکنیم.

Training with adam:

Training on device: cuda

```
Epoch 1/100: 100%|██████████| 184/184 [00:48<00:00, 3.77it/s]
[Epoch 1] Loss: 0.7038 | IoU: 0.2481 | Dice: 2.5568 | Acc: 65.77%
✓ New best IoU: 0.2481 (model saved)
Epoch 2/100: 100%|██████████| 184/184 [00:46<00:00, 3.92it/s]
[Epoch 2] Loss: 0.5101 | IoU: 0.3185 | Dice: 3.4115 | Acc: 73.85%
✓ New best IoU: 0.3185 (model saved)
Epoch 3/100: 100%|██████████| 184/184 [00:49<00:00, 3.69it/s]
[Epoch 3] Loss: 0.4522 | IoU: 0.3937 | Dice: 4.6431 | Acc: 81.58%
✓ New best IoU: 0.3937 (model saved)
Epoch 4/100: 100%|██████████| 184/184 [00:47<00:00, 3.84it/s]
[Epoch 4] Loss: 0.4038 | IoU: 0.4153 | Dice: 5.2416 | Acc: 83.90%
✓ New best IoU: 0.4153 (model saved)
Epoch 5/100: 100%|██████████| 184/184 [00:47<00:00, 3.90it/s]
[Epoch 5] Loss: 0.3772 | IoU: 0.4213 | Dice: 4.9318 | Acc: 83.45%
✓ New best IoU: 0.4213 (model saved)
Epoch 6/100: 100%|██████████| 184/184 [00:48<00:00, 3.80it/s]
[Epoch 6] Loss: 0.3525 | IoU: 0.3969 | Dice: 4.6449 | Acc: 80.68%
No improvement. EarlyStopping counter: 1/10
Epoch 7/100: 100%|██████████| 184/184 [00:47<00:00, 3.91it/s]
[Epoch 7] Loss: 0.3332 | IoU: 0.4653 | Dice: 5.3892 | Acc: 86.83%
✓ New best IoU: 0.4653 (model saved)
Epoch 8/100: 100%|██████████| 184/184 [00:48<00:00, 3.80it/s]
[Epoch 8] Loss: 0.3175 | IoU: 0.4535 | Dice: 5.2968 | Acc: 86.40%
No improvement. EarlyStopping counter: 1/10
Epoch 9/100: 100%|██████████| 184/184 [00:46<00:00, 3.93it/s]
[Epoch 9] Loss: 0.3021 | IoU: 0.4413 | Dice: 5.1578 | Acc: 83.78%
No improvement. EarlyStopping counter: 2/10
Epoch 10/100: 100%|██████████| 184/184 [00:48<00:00, 3.78it/s]
[Epoch 10] Loss: 0.2940 | IoU: 0.4977 | Dice: 5.9157 | Acc: 87.27%
✓ New best IoU: 0.4977 (model saved)
Epoch 11/100: 100%|██████████| 184/184 [00:48<00:00, 3.76it/s]
[Epoch 11] Loss: 0.2795 | IoU: 0.4725 | Dice: 5.2199 | Acc: 87.33%
No improvement. EarlyStopping counter: 1/10
Epoch 12/100: 100%|██████████| 184/184 [00:47<00:00, 3.88it/s]
[Epoch 12] Loss: 0.2717 | IoU: 0.4810 | Dice: 5.4048 | Acc: 85.88%
No improvement. EarlyStopping counter: 2/10
Epoch 13/100: 100%|██████████| 184/184 [00:48<00:00, 3.81it/s]
[Epoch 13] Loss: 0.2634 | IoU: 0.5067 | Dice: 5.4665 | Acc: 87.89%
✓ New best IoU: 0.5067 (model saved)
Epoch 14/100: 100%|██████████| 184/184 [00:47<00:00, 3.91it/s]
[Epoch 14] Loss: 0.2522 | IoU: 0.4760 | Dice: 5.3810 | Acc: 86.75%
No improvement. EarlyStopping counter: 1/10
Epoch 15/100: 100%|██████████| 184/184 [00:47<00:00, 3.87it/s]
[Epoch 15] Loss: 0.2419 | IoU: 0.5083 | Dice: 5.5964 | Acc: 87.79%
✓ New best IoU: 0.5083 (model saved)
Epoch 16/100: 100%|██████████| 184/184 [00:48<00:00, 3.80it/s]
[Epoch 16] Loss: 0.2384 | IoU: 0.4937 | Dice: 4.9255 | Acc: 88.66%
No improvement. EarlyStopping counter: 1/10
Epoch 17/100: 100%|██████████| 184/184 [00:46<00:00, 3.95it/s]
[Epoch 17] Loss: 0.2363 | IoU: 0.5105 | Dice: 5.1884 | Acc: 88.99%
✓ New best IoU: 0.5105 (model saved)
Epoch 18/100: 100%|██████████| 184/184 [00:48<00:00, 3.82it/s]
[Epoch 18] Loss: 0.2273 | IoU: 0.5265 | Dice: 5.6220 | Acc: 88.84%
✓ New best IoU: 0.5265 (model saved)
Epoch 19/100: 100%|██████████| 184/184 [00:46<00:00, 3.94it/s]
[Epoch 19] Loss: 0.2263 | IoU: 0.5209 | Dice: 5.4697 | Acc: 88.32%
No improvement. EarlyStopping counter: 1/10
Epoch 20/100: 100%|██████████| 184/184 [00:48<00:00, 3.81it/s]
[Epoch 20] Loss: 0.2243 | IoU: 0.5198 | Dice: 5.5821 | Acc: 88.60%
No improvement. EarlyStopping counter: 2/10
Epoch 21/100: 100%|██████████| 184/184 [00:46<00:00, 3.94it/s]
[Epoch 21] Loss: 0.2140 | IoU: 0.5228 | Dice: 5.6813 | Acc: 88.53%
No improvement. EarlyStopping counter: 3/10
Epoch 22/100: 100%|██████████| 184/184 [00:47<00:00, 3.89it/s]
[Epoch 22] Loss: 0.2105 | IoU: 0.5100 | Dice: 5.7641 | Acc: 88.70%
No improvement. EarlyStopping counter: 4/10
Epoch 23/100: 100%|██████████| 184/184 [00:47<00:00, 3.85it/s]
[Epoch 23] Loss: 0.2109 | IoU: 0.5016 | Dice: 5.4465 | Acc: 88.70%
No improvement. EarlyStopping counter: 5/10
Epoch 24/100: 100%|██████████| 184/184 [00:46<00:00, 3.95it/s]
[Epoch 24] Loss: 0.2032 | IoU: 0.5331 | Dice: 5.6546 | Acc: 89.27%
✓ New best IoU: 0.5331 (model saved)
Epoch 25/100: 100%|██████████| 184/184 [00:48<00:00, 3.78it/s]
[Epoch 25] Loss: 0.2009 | IoU: 0.5194 | Dice: 5.4251 | Acc: 89.65%
No improvement. EarlyStopping counter: 1/10
Epoch 26/100: 100%|██████████| 184/184 [00:46<00:00, 3.92it/s]
[Epoch 26] Loss: 0.1968 | IoU: 0.5429 | Dice: 5.8670 | Acc: 89.02%
✓ New best IoU: 0.5429 (model saved)
Epoch 27/100: 100%|██████████| 184/184 [00:49<00:00, 3.74it/s]
[Epoch 27] Loss: 0.1943 | IoU: 0.5254 | Dice: 5.4985 | Acc: 89.48%
No improvement. EarlyStopping counter: 1/10
Epoch 28/100: 100%|██████████| 184/184 [00:48<00:00, 3.80it/s]
[Epoch 28] Loss: 0.1899 | IoU: 0.5111 | Dice: 5.4755 | Acc: 89.13%
No improvement. EarlyStopping counter: 2/10
Epoch 29/100: 100%|██████████| 184/184 [00:47<00:00, 3.88it/s]
[Epoch 29] Loss: 0.1909 | IoU: 0.5186 | Dice: 5.5533 | Acc: 89.15%
No improvement. EarlyStopping counter: 3/10
Epoch 30/100: 100%|██████████| 184/184 [00:48<00:00, 3.79it/s]
[Epoch 30] Loss: 0.1872 | IoU: 0.5400 | Dice: 5.4630 | Acc: 89.85%
No improvement. EarlyStopping counter: 4/10
Epoch 31/100: 100%|██████████| 184/184 [00:47<00:00, 3.91it/s]
[Epoch 31] Loss: 0.1804 | IoU: 0.5243 | Dice: 5.5307 | Acc: 89.39%
No improvement. EarlyStopping counter: 5/10
Epoch 32/100: 100%|██████████| 184/184 [00:48<00:00, 3.79it/s]
```

```

[Epoch 32] Loss: 0.1808 | IoU: 0.5166 | Dice: 5.5485 | Acc: 88.45%
No improvement. EarlyStopping counter: 6/10
Epoch 33/100: 100%|██████████| 184/184 [00:48<00:00, 3.80it/s]
[Epoch 33] Loss: 0.1771 | IoU: 0.5175 | Dice: 5.4791 | Acc: 89.24%
No improvement. EarlyStopping counter: 7/10
Epoch 34/100: 100%|██████████| 184/184 [00:47<00:00, 3.89it/s]
[Epoch 34] Loss: 0.1764 | IoU: 0.5298 | Dice: 5.5895 | Acc: 88.50%
No improvement. EarlyStopping counter: 8/10
Epoch 35/100: 100%|██████████| 184/184 [00:48<00:00, 3.83it/s]
[Epoch 35] Loss: 0.1756 | IoU: 0.5215 | Dice: 5.3906 | Acc: 89.27%
No improvement. EarlyStopping counter: 9/10
Epoch 36/100: 100%|██████████| 184/184 [00:48<00:00, 3.81it/s]
[Epoch 36] Loss: 0.1729 | IoU: 0.5301 | Dice: 5.4748 | Acc: 89.69%
No improvement. EarlyStopping counter: 10/10
Early stopping triggered. Best IoU: 0.5429

```

Training with sgd:

Training on device: cuda

```

Epoch 1/100: 100%|██████████| 184/184 [00:46<00:00, 3.93it/s]
[Epoch 1] Loss: 0.9712 | IoU: 0.2503 | Dice: 4.2543 | Acc: 67.26%
✓ New best IoU: 0.2503 (model saved)
Epoch 2/100: 100%|██████████| 184/184 [00:50<00:00, 3.68it/s]
[Epoch 2] Loss: 0.6439 | IoU: 0.3013 | Dice: 3.2232 | Acc: 73.04%
✓ New best IoU: 0.3013 (model saved)
Epoch 3/100: 100%|██████████| 184/184 [00:48<00:00, 3.81it/s]
[Epoch 3] Loss: 0.5663 | IoU: 0.3379 | Dice: 3.7164 | Acc: 78.28%
✓ New best IoU: 0.3379 (model saved)
Epoch 4/100: 100%|██████████| 184/184 [00:47<00:00, 3.88it/s]
[Epoch 4] Loss: 0.5180 | IoU: 0.3308 | Dice: 4.4531 | Acc: 77.86%
No improvement. EarlyStopping counter: 1/10
Epoch 5/100: 100%|██████████| 184/184 [00:48<00:00, 3.78it/s]
[Epoch 5] Loss: 0.4776 | IoU: 0.3124 | Dice: 3.2069 | Acc: 74.27%
No improvement. EarlyStopping counter: 2/10
Epoch 6/100: 100%|██████████| 184/184 [00:47<00:00, 3.88it/s]
[Epoch 6] Loss: 0.4539 | IoU: 0.3554 | Dice: 4.7899 | Acc: 80.09%
✓ New best IoU: 0.3554 (model saved)
Epoch 7/100: 100%|██████████| 184/184 [00:48<00:00, 3.80it/s]
[Epoch 7] Loss: 0.4272 | IoU: 0.3373 | Dice: 3.9795 | Acc: 76.61%
No improvement. EarlyStopping counter: 1/10
Epoch 8/100: 100%|██████████| 184/184 [00:49<00:00, 3.74it/s]
[Epoch 8] Loss: 0.4069 | IoU: 0.3946 | Dice: 5.2412 | Acc: 82.61%
✓ New best IoU: 0.3946 (model saved)
Epoch 9/100: 100%|██████████| 184/184 [00:47<00:00, 3.87it/s]
[Epoch 9] Loss: 0.3882 | IoU: 0.3840 | Dice: 4.5635 | Acc: 81.60%
No improvement. EarlyStopping counter: 1/10
Epoch 10/100: 100%|██████████| 184/184 [00:49<00:00, 3.68it/s]
[Epoch 10] Loss: 0.3742 | IoU: 0.3895 | Dice: 4.8182 | Acc: 81.66%
No improvement. EarlyStopping counter: 2/10
Epoch 11/100: 100%|██████████| 184/184 [00:49<00:00, 3.69it/s]

```

```

[Epoch 11] Loss: 0.3650 | IoU: 0.3932 | Dice: 4.9157 | Acc: 82.02%
No improvement. EarlyStopping counter: 3/10
Epoch 12/100: 100%|██████████| 184/184 [00:48<00:00, 3.77it/s]
[Epoch 12] Loss: 0.3522 | IoU: 0.3915 | Dice: 4.7608 | Acc: 81.71%
No improvement. EarlyStopping counter: 4/10
Epoch 13/100: 100%|██████████| 184/184 [00:48<00:00, 3.78it/s]
[Epoch 13] Loss: 0.3447 | IoU: 0.4079 | Dice: 4.7621 | Acc: 82.80%
✓ New best IoU: 0.4079 (model saved)
Epoch 14/100: 100%|██████████| 184/184 [00:49<00:00, 3.71it/s]
[Epoch 14] Loss: 0.3353 | IoU: 0.3911 | Dice: 4.8533 | Acc: 81.23%
No improvement. EarlyStopping counter: 1/10
Epoch 15/100: 100%|██████████| 184/184 [00:48<00:00, 3.76it/s]
[Epoch 15] Loss: 0.3277 | IoU: 0.4344 | Dice: 5.4529 | Acc: 84.82%
✓ New best IoU: 0.4344 (model saved)
Epoch 16/100: 100%|██████████| 184/184 [00:47<00:00, 3.87it/s]
[Epoch 16] Loss: 0.3215 | IoU: 0.4102 | Dice: 4.7159 | Acc: 83.54%
No improvement. EarlyStopping counter: 1/10
Epoch 17/100: 100%|██████████| 184/184 [00:49<00:00, 3.70it/s]
[Epoch 17] Loss: 0.3114 | IoU: 0.3864 | Dice: 4.5050 | Acc: 80.86%
No improvement. EarlyStopping counter: 2/10
Epoch 18/100: 100%|██████████| 184/184 [00:48<00:00, 3.82it/s]
[Epoch 18] Loss: 0.3053 | IoU: 0.4103 | Dice: 4.8132 | Acc: 83.89%
No improvement. EarlyStopping counter: 3/10
Epoch 19/100: 100%|██████████| 184/184 [00:48<00:00, 3.81it/s]
[Epoch 19] Loss: 0.3021 | IoU: 0.4173 | Dice: 5.2338 | Acc: 83.55%
No improvement. EarlyStopping counter: 4/10
Epoch 20/100: 100%|██████████| 184/184 [00:49<00:00, 3.71it/s]
[Epoch 20] Loss: 0.2961 | IoU: 0.4108 | Dice: 4.9727 | Acc: 83.68%
No improvement. EarlyStopping counter: 5/10
Epoch 21/100: 100%|██████████| 184/184 [00:47<00:00, 3.86it/s]
[Epoch 21] Loss: 0.2898 | IoU: 0.4369 | Dice: 5.2288 | Acc: 85.13%
✓ New best IoU: 0.4369 (model saved)
Epoch 22/100: 100%|██████████| 184/184 [00:48<00:00, 3.77it/s]
[Epoch 22] Loss: 0.2833 | IoU: 0.4258 | Dice: 5.0443 | Acc: 84.49%
No improvement. EarlyStopping counter: 1/10
Epoch 23/100: 100%|██████████| 184/184 [00:49<00:00, 3.75it/s]
[Epoch 23] Loss: 0.2832 | IoU: 0.4299 | Dice: 5.2491 | Acc: 84.27%
No improvement. EarlyStopping counter: 2/10
Epoch 24/100: 100%|██████████| 184/184 [00:47<00:00, 3.88it/s]
[Epoch 24] Loss: 0.2759 | IoU: 0.4069 | Dice: 4.8258 | Acc: 82.50%
No improvement. EarlyStopping counter: 3/10
Epoch 25/100: 100%|██████████| 184/184 [00:49<00:00, 3.72it/s]
[Epoch 25] Loss: 0.2760 | IoU: 0.4275 | Dice: 4.9286 | Acc: 84.56%
No improvement. EarlyStopping counter: 4/10
Epoch 26/100: 100%|██████████| 184/184 [00:48<00:00, 3.83it/s]
[Epoch 26] Loss: 0.2695 | IoU: 0.4304 | Dice: 5.3279 | Acc: 84.64%
No improvement. EarlyStopping counter: 5/10
Epoch 27/100: 100%|██████████| 184/184 [00:46<00:00, 3.95it/s]
[Epoch 27] Loss: 0.2691 | IoU: 0.4381 | Dice: 5.2596 | Acc: 85.09%
✓ New best IoU: 0.4381 (model saved)

```

```

Epoch 28/100: 100%|██████████| 184/184
[00:47<00:00, 3.83it/s]
[T] [Epoch 28] Loss: 0.2658 | IoU: 0.4239 |
Dice: 4.9805 | Acc: 84.14%
No improvement. EarlyStopping counter: 1/10
Epoch 29/100: 100%|██████████| 184/184
[00:47<00:00, 3.91it/s]
[T] [Epoch 29] Loss: 0.2592 | IoU: 0.4323 |
Dice: 5.1638 | Acc: 84.47%
No improvement. EarlyStopping counter: 2/10
Epoch 30/100: 100%|██████████| 184/184
[00:47<00:00, 3.85it/s]
[T] [Epoch 30] Loss: 0.2576 | IoU: 0.4282 |
Dice: 5.1783 | Acc: 84.23%
No improvement. EarlyStopping counter: 3/10
Epoch 31/100: 100%|██████████| 184/184
[00:47<00:00, 3.84it/s]
[T] [Epoch 31] Loss: 0.2550 | IoU: 0.4329 |
Dice: 5.0870 | Acc: 84.58%
No improvement. EarlyStopping counter: 4/10
Epoch 32/100: 100%|██████████| 184/184
[00:46<00:00, 3.92it/s]
[T] [Epoch 32] Loss: 0.2517 | IoU: 0.4468 |
Dice: 5.3880 | Acc: 85.27%
✓ New best IoU: 0.4468 (model saved)
Epoch 33/100: 100%|██████████| 184/184
[00:48<00:00, 3.83it/s]
[T] [Epoch 33] Loss: 0.2477 | IoU: 0.4487 |
Dice: 5.2605 | Acc: 85.43%
✓ New best IoU: 0.4487 (model saved)
Epoch 34/100: 100%|██████████| 184/184
[00:47<00:00, 3.90it/s]
[T] [Epoch 34] Loss: 0.2483 | IoU: 0.4401 |
Dice: 5.1674 | Acc: 84.85%
No improvement. EarlyStopping counter: 1/10
Epoch 35/100: 100%|██████████| 184/184
[00:46<00:00, 3.95it/s]
[T] [Epoch 35] Loss: 0.2476 | IoU: 0.4464 |
Dice: 5.1760 | Acc: 85.78%
No improvement. EarlyStopping counter: 2/10
Epoch 36/100: 100%|██████████| 184/184
[00:48<00:00, 3.83it/s]
[T] [Epoch 36] Loss: 0.2436 | IoU: 0.4403 |
Dice: 5.1274 | Acc: 84.56%
No improvement. EarlyStopping counter: 3/10
Epoch 37/100: 100%|██████████| 184/184
[00:46<00:00, 3.95it/s]
[T] [Epoch 37] Loss: 0.2405 | IoU: 0.4444 |
Dice: 5.1835 | Acc: 85.13%
No improvement. EarlyStopping counter: 4/10
Epoch 38/100: 100%|██████████| 184/184
[00:47<00:00, 3.84it/s]
[T] [Epoch 38] Loss: 0.2405 | IoU: 0.4521 |
Dice: 5.2674 | Acc: 85.61%
✓ New best IoU: 0.4521 (model saved)
Epoch 39/100: 100%|██████████| 184/184
[00:48<00:00, 3.79it/s]
[T] [Epoch 39] Loss: 0.2369 | IoU: 0.4380 |
Dice: 5.2280 | Acc: 85.36%
No improvement. EarlyStopping counter: 1/10
Epoch 40/100: 100%|██████████| 184/184
[00:47<00:00, 3.90it/s]
[T] [Epoch 40] Loss: 0.2342 | IoU: 0.4412 |
Dice: 5.4140 | Acc: 84.63%
No improvement. EarlyStopping counter: 2/10
Epoch 41/100: 100%|██████████| 184/184
[00:49<00:00, 3.73it/s]
[T] [Epoch 41] Loss: 0.2355 | IoU: 0.4308 |
Dice: 5.1424 | Acc: 83.81%
No improvement. EarlyStopping counter: 3/10
Epoch 42/100: 100%|██████████| 184/184
[00:49<00:00, 3.74it/s]
[T] [Epoch 42] Loss: 0.2332 | IoU: 0.4385 |
Dice: 5.1237 | Acc: 85.02%
No improvement. EarlyStopping counter: 4/10
Epoch 43/100: 100%|██████████| 184/184
[00:47<00:00, 3.84it/s]
[T] [Epoch 43] Loss: 0.2298 | IoU: 0.4432 |
Dice: 5.2454 | Acc: 85.33%
No improvement. EarlyStopping counter: 5/10
Epoch 44/100: 100%|██████████| 184/184
[00:48<00:00, 3.76it/s]
[T] [Epoch 44] Loss: 0.2296 | IoU: 0.4435 |
Dice: 5.2505 | Acc: 85.24%
No improvement. EarlyStopping counter: 6/10
Epoch 45/100: 100%|██████████| 184/184
[00:49<00:00, 3.73it/s]
[T] [Epoch 45] Loss: 0.2282 | IoU: 0.4475 |
Dice: 5.2334 | Acc: 84.88%
No improvement. EarlyStopping counter: 7/10
Epoch 46/100: 100%|██████████| 184/184
[00:46<00:00, 3.92it/s]
[T] [Epoch 46] Loss: 0.2269 | IoU: 0.4461 |
Dice: 5.1581 | Acc: 85.44%
No improvement. EarlyStopping counter: 8/10
Epoch 47/100: 100%|██████████| 184/184
[00:47<00:00, 3.84it/s]
[T] [Epoch 47] Loss: 0.2253 | IoU: 0.4559 |
Dice: 5.3139 | Acc: 85.65%
✓ New best IoU: 0.4559 (model saved)
Epoch 48/100: 100%|██████████| 184/184
[00:48<00:00, 3.80it/s]
[T] [Epoch 48] Loss: 0.2239 | IoU: 0.4477 |
Dice: 5.2810 | Acc: 85.60%
No improvement. EarlyStopping counter: 1/10
Epoch 49/100: 100%|██████████| 184/184
[00:46<00:00, 3.92it/s]
[T] [Epoch 49] Loss: 0.2204 | IoU: 0.4380 |
Dice: 5.1209 | Acc: 84.10%
No improvement. EarlyStopping counter: 2/10
Epoch 50/100: 100%|██████████| 184/184
[00:47<00:00, 3.84it/s]
[T] [Epoch 50] Loss: 0.2216 | IoU: 0.4471 |
Dice: 5.2973 | Acc: 85.03%
No improvement. EarlyStopping counter: 3/10
Epoch 51/100: 100%|██████████| 184/184
[00:48<00:00, 3.80it/s]
[T] [Epoch 51] Loss: 0.2208 | IoU: 0.4504 |
Dice: 5.3127 | Acc: 84.91%
No improvement. EarlyStopping counter: 4/10
Epoch 52/100: 100%|██████████| 184/184
[00:46<00:00, 3.93it/s]
[T] [Epoch 52] Loss: 0.2177 | IoU: 0.4515 |
Dice: 5.2111 | Acc: 85.57%
No improvement. EarlyStopping counter: 5/10
Epoch 53/100: 100%|██████████| 184/184
[00:48<00:00, 3.83it/s]
[T] [Epoch 53] Loss: 0.2173 | IoU: 0.4537 |
Dice: 5.4085 | Acc: 85.30%
No improvement. EarlyStopping counter: 6/10
Epoch 54/100: 100%|██████████| 184/184
[00:48<00:00, 3.83it/s]
[T] [Epoch 54] Loss: 0.2141 | IoU: 0.4486 |
Dice: 5.3567 | Acc: 84.44%
No improvement. EarlyStopping counter: 7/10
Epoch 55/100: 100%|██████████| 184/184
[00:46<00:00, 3.92it/s]
[T] [Epoch 55] Loss: 0.2147 | IoU: 0.4410 |
Dice: 5.1201 | Acc: 85.06%
No improvement. EarlyStopping counter: 8/10
Epoch 56/100: 100%|██████████| 184/184
[00:48<00:00, 3.80it/s]
[T] [Epoch 56] Loss: 0.2130 | IoU: 0.4544 |
Dice: 5.2499 | Acc: 85.73%
No improvement. EarlyStopping counter: 9/10
Epoch 57/100: 100%|██████████| 184/184
[00:46<00:00, 3.92it/s]
[T] [Epoch 57] Loss: 0.2120 | IoU: 0.4515 |
Dice: 5.3102 | Acc: 85.18%
No improvement. EarlyStopping counter: 10/10
Early stopping triggered. Best IoU: 0.4559

```

با توجه به این لگ ها IoU های نهایی مدلی که با adam آموزش دیده 54.29 % و مدلی که با sgd آموزش

دیده، 43.85 % است. ایپاک هایی که مدل ها به این دقت ها رسیده اند در زیر آورده شده:

: Adam

```
[Epoch 26] Loss: 0.1968 | IoU: 0.5429 | Dice: 5.8670 | Acc: 89.02%
✓ New best IoU: 0.5429 (model saved)
Epoch 27/100: 100% | ██████████ | 184/184 [00:49<00:00, 3.74it/s]
```

: Sgd

```
[Epoch 38] Loss: 0.2405 | IoU: 0.4521 | Dice: 5.2674 | Acc: 85.61%
✓ New best IoU: 0.4521 (model saved)
Epoch 39/100: 100% | ██████████ | 184/184 [00:48<00:00, 3.79it/s]
```

با توجه به این مقادیر در مدلی که توسط adam آموزش دیده، معیار های IoU و Dice به بالاتر از 50% رسیده اند.

۱-۶. ارزیابی مدل

مقادیر متریک های مورد بررسی روی داده های تست به این ترتیب است:

adam:

```
Test IoU: 0.3846 | Dice: 5.2336 | Accuracy: 82.92%
```

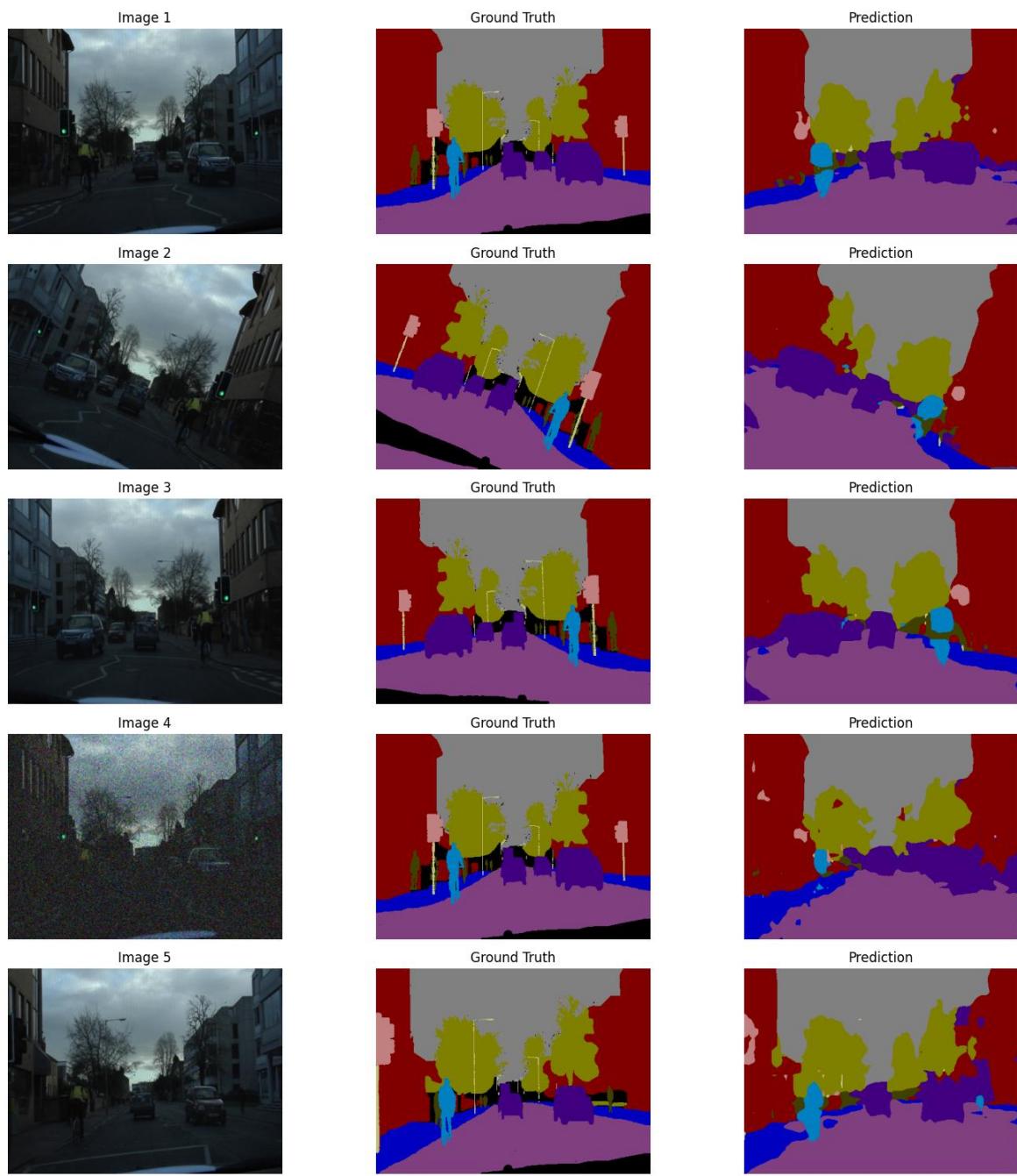
sgd:

```
Test IoU: 0.3396 | Dice: 4.7684 | Accuracy: 79.35%
```

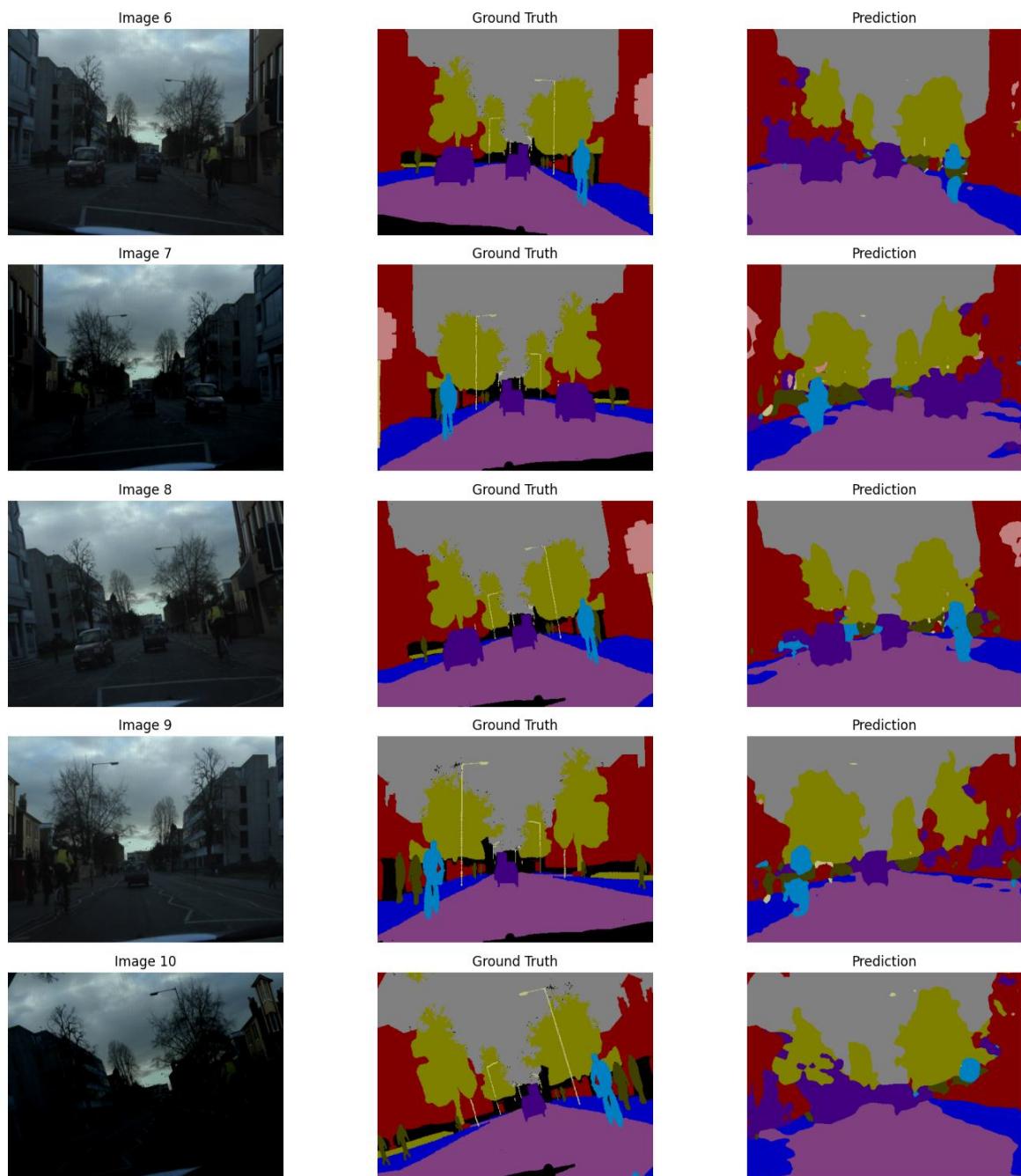
با توجه به این موارد می توانیم بینیم مدل روی داده های تست عملکرد ضعیف تری داشته (که طبیعی است) ولی همچنان اینجا هم بهینه ساز adam عملکرد بهتری از خود نشان داده است.

در ادامه چند نمونه از تصاویر و خروجی مدل برای آنها را بررسی میکنیم.

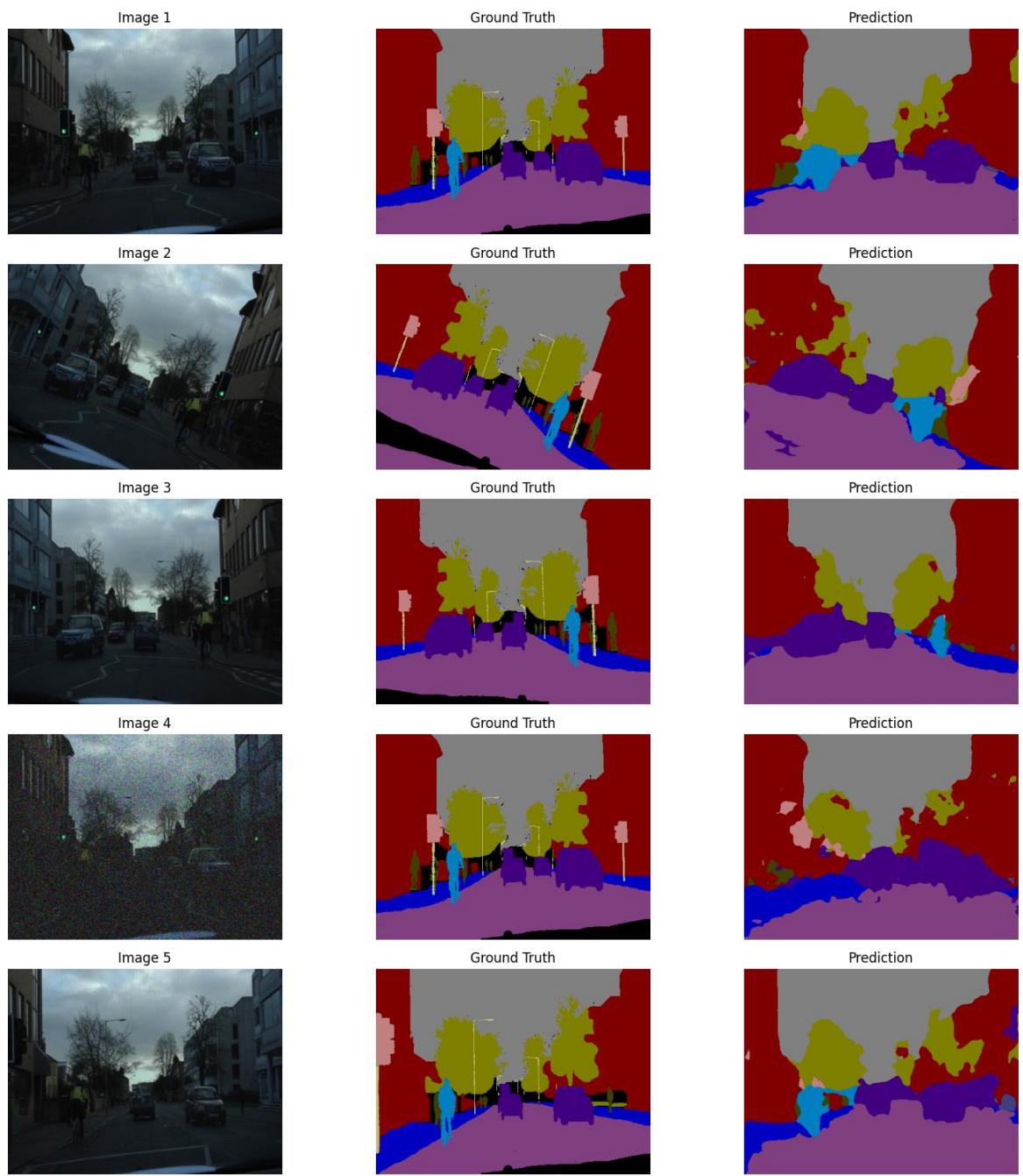
با توجه به شکل های ۶ تا ۹ مدل روی داده هایی که چرخش یافته اند ی به صورت افقی قرینه شده اند عملکرد خوب و قابل قبولی داشته است ولی روی داده هایی که نویز روی آنها سوار شده و یا شدت نور آنها تغییر یافته است، مدل عملکرد ضعیف تری داشته است. بنابراین می توانیم بگوییم در شرایط واقعی کیفیت دوربین و شدت نور محیط پارامتر بسیار تاثیرگذاری بر روی دقت مدل است. ولی به طور کلی می توان گفت مدل عملکرد بدی نداشته (مخصوصاً به عنوان یک مدل real time object) و های تصاویر خروجی قابل تفکیک و تشخیص هستند.



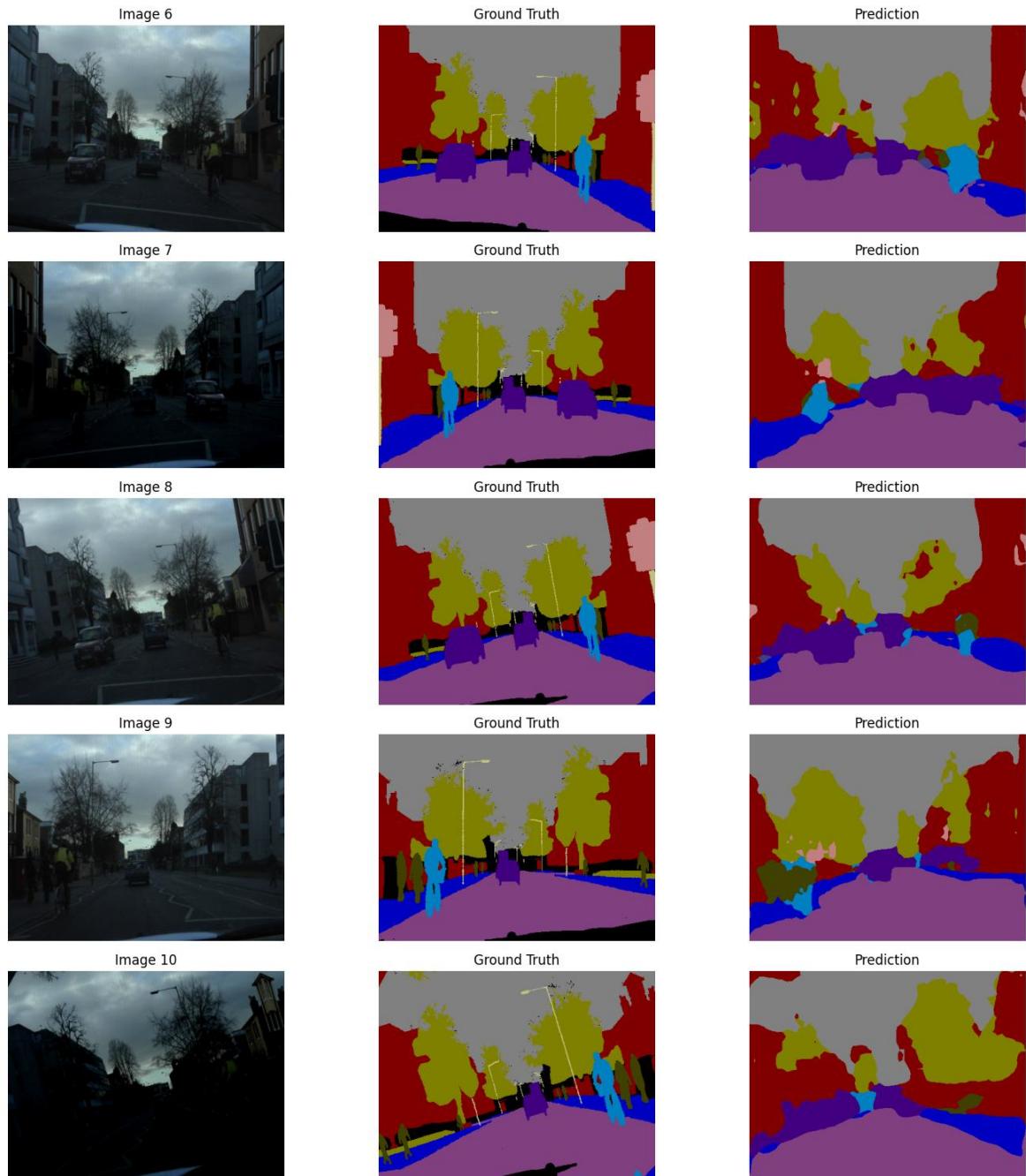
شکل ۶ : تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز adam) تصاویر ۱ تا ۵



شکل ۷ : تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز adam) تصاویر ۶ تا ۱۰



شکل ۸ : تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز sgd) تصاویر ۱ تا ۵



شکل ۹ : تصاویر نمونه پیش‌بینی شده توسط مدل (با بهینه ساز SGD) تصاویر ۶ تا ۱۰

پرسش ۲ – Oriented R-CNN برای تشخیص اشیا

بخش اول: سوالات نظری

مقدمه

همانطور که در این سرفصل از درس دیدیم، یکی از کاربردهای گسترده شبکه‌های عصبی موضوع Object detection and segmentation می‌باشد و مدل‌های مختلفی نیز برای این کاربرد معرفی شده‌اند. مبنای کار یک سری از این مدل‌ها که با نام two-stage detector شناخته می‌شوند، پیدا کردن مجموعه ای از Region Proposal ها در مرحله اول، و بررسی جدگانه هریک از آنها در مرحله بعد است. همانطور که دیده بودیم اکثر شبکه‌های معرفی شده تنها Proposal های افقی را از تصویر استخراج می‌کردند؛ حال آنکه در بسیاری از کاربردها مانند تصویربرداری هوایی، میخواهیم اشیایی که با زاویه‌ای چرخیده‌اند را نیز به درستی مکان‌یابی و مرزهای دقیقش را مشخص کرده و همچنین طبقه‌بندی کنیم. در این شرایط استفاده از همان Horizontal proposal ها می‌تواند باعث کاهش دقت و افزایش خطای شبکه شود. برای همین از شبکه‌های Oriented object detection استفاده می‌کنند.

این شبکه‌ها معمولاً بر مبنای تشکیل دادن Oriented proposal ها (که دارای زوایه‌ای در صفحه هستند) در مرحله اول، و تحلیل و طبقه‌بندی آن‌ها در مرحله بعد می‌باشند.

درک مفهومی

الف: انگیزه اصلی توسعه Oriented R-CNN

همانطور که در مقدمه ذکر شد، چالش اصلی در شبکه‌های Oriented object detector بدبست آوردن Oriented proposal ها می‌باشد که از لحاظ محاسباتی کار پرهزینه و وقت‌گیری است و همین باعث می‌شود این عملیات Bottleneck های شبکه باشد. انگیزه اصلی توسعه این شبکه، بدبست آوردن یک oriented region proposal network (oriented RPN) هست که با هزینه کم و به صورت مستقیم Oriented proposal ها را بدبست آورد.

این مدل نسبت به روش‌های قدیمی برتری‌هایی دارد. برای مثال با استفاده Horizontal box های متداول نمیتوانیم به درستی موقعیت اشیای زاویه دار را تشخیص دهیم؛ زیرا این باکس‌ها می‌توانند شامل بخش زیادی پس‌زمینه باشند یا حتی شامل چندین Oriented Object باشند که از هم تفکیک نشوند. یکی دیگر از روش‌های اولیه، استفاده از تعداد زیادی Anchor box با زاویه‌های متفاوت می‌باشد ولی بدلیل تعداد بالای Proposal ها، حجم محاسبات و استفاده از حافظه بسیار بالایی دارد. برای رفع این مشکل

شبکه ROI transformer معرفی شده که از Rotated RoIs به جهت یادگیری Horizontal RoIs استفاده می‌کند. با این حال این شبکه نیز بدلیل داشتن لایه‌های Fully Connected بزرگ در یادگیری RoIs، حجم محاسبات خیلی زیادی دارد.

ب: مزایای استفاده از نمایش Midpoint offset

در این مقاله برای Represent کردن anchor box‌ها از نمایش Midpoint offset استفاده شده است. در این نمایش، جعبه با ۶ پارامتر مشخص می‌شود (طول و عرض باکس افقی محاط، مختصات مرکز آن، و فاصله دو راس باکس چرخیده شده از نقاط میانی یال‌های بالا و راست باکس افقی اولیه). با استفاده از این ۶ پارامتر می‌توان مختصات هر ۴ راس Rotated box را بدست آورد (با روابط ریاضی ای که در مقاله ذکر شده است). مزیت این روش این است که از همان مکانیسم Regression که برای Horizontal box‌ها استفاده می‌کردیم بهره می‌گیرید و با تعیین ۲ آفست، موقعیت Oriented box‌ها را می‌تواند پیش‌بینی کند (از یک شبکه CNN سبک برای این کار بهره می‌گیرد).

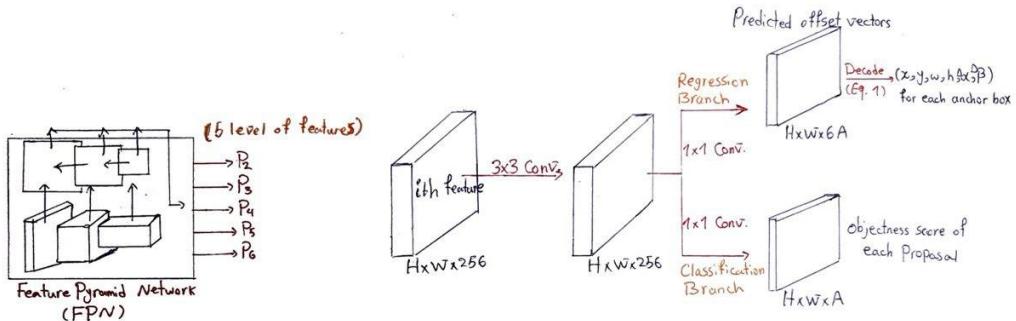
همچنین با اعمال این نمایش، شاخه Regression بخش RPN، ۶ آفست (برای هریک از پارامترهای نمایش) بدست می‌آورد و با اعمال روابط ۱ در مقاله، می‌توان باکس خارجی را به نحو خوبی Adjust کرد تا شامل Oriented Object مورد نظر بشود.

برای مثال در تشخیص اجسام باریک که به صورت قطری در تصویر قرار گرفته‌اند، بخش زیادی از باکس‌های Horizontal را پس‌زمینه و اجسام دیگر تشکیل می‌دهند. ولی با نمایش Midpoint offset، مرازهای دقیق جسم را می‌توان مشخص کرد. همچنین دیدیم شبکه‌های ROI transformer هم که از همان نمایش سنتی Oriented box استفاده می‌کردند، برای یادگیری Overhead Oriented box‌ها زیادی داشتند.

اجزای مدل

الف: معماری Oriented RPN

مطابق مقاله، Oriented RPN بخش اول از مدل Oriented R-CNN ارائه شده می‌باشد که وظیفه تشکیل Oriented proposal را بر عهده دارد. شکل زیر معماری این بخش را نشان می‌دهد. در ادامه جزئیات هر بخش را توضیح می‌دهیم.



شکل ۱۰۰: معماری RPN

برای استخراج ویژگی‌ها از یک Backbone ResNet FPN که شبکه Feature Pyramid Network مانند استفاده می‌کند. از ۵ لول از feature map های این شبکه استفاده می‌کنیم و به هر کدام به صورت جداگانه، یک head مطابق تصویر رسم شده متصل می‌کنیم.

در هر نقطه از feature map های ورودی، ۳ Anchor Box با نسبت های $\{2:1, 1:2, 1:1\}$ را اعمال می‌کنیم (سایز این باکس‌ها برای هر کدام از ۵ feature map ورودی، در مقاله ذکر شده است).

ابتدا یک لایه 3×3 Convolutional به feature map ورودی اعمال می‌کنیم. سپس ۲ لایه Convolutional به صورت موازی به خروجی این لایه اعمال می‌شوند.

لایه بالایی Regression Branch می‌باشد که وظیفه اش بدست آوردن آفست‌های ۶ پارامتر نمایش Midpoint offset برای هر یک از anchor box های اعمال شده در هر نقطه از تصویر می‌باشد. به همین دلیل خروجی این لایه تعداد ۶A کانال دارد. (A تعداد Anchore Box ها در هر موقعیت مکانی پیکسل می‌باشد که در این پیاده‌سازی ۳ هست). در مرحله بعد برای هر کدام از Anchor box ها در هر موقعیت مکانی، با استفاده از رابطه ۱ در مقاله و داشتن طول و عرض آن باکس و آفست‌های پیش‌بینی شده، نمایش آن باکس را بدست می‌اوریم.

لایه پایینی Classification Branch می‌باشد که وظیفه اش پیش‌بینی Objectness score برای هر Anchor Box می‌باشد، که احتمال این که شامل یک شی باشد و یا تنها تصویر پس‌زمینه هست را به ما می‌دهد. خروجی این لایه A کانال دارد؛ در هر موقعیت مکانی A باکس داریم که باید شاخص Objectness score را برای هر یک جداگانه مشخص کنیم.

تفاوت‌های این مدل با RPN سنتی به صورت زیر است:

در RPN سنتی تنها ۴ پارامتر برای پیش‌بینی باکس‌های افقی پیش‌بینی می‌کنیم، ولی در این مدل با استفاده از ۶ پارامتر و به کمک نمایش Modpoint offset، توانستیم Oriented box ها را نیز پیش‌بینی کنیم؛ حال آنکه در اینجا نیز از Anchor box های افقی استفاده می‌کنیم.

ب: نحوه فرمول بندی تابع هزینه

تابع هزینه در Oriented RPN به صورت زیر تعریف می‌شود:

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

هدف از تعریف آن به صورت بالا این است که هر دو شاخه Classification و Regression که در بخش قبل توضیح دادیم را لحاظ کرده و همزمان بهینه کند.

در رابطه بالا، نیز تعداد سمپل‌های یک mini-batch اعمال شده می‌باشد (به صورت پیش فرض ۲۵۶ است).

برای تعریف رابطه بالا، به هریک از Anchor Box‌ها پارامتر p را نسبت می‌دهند که در صورت صفر بودن به معنای Negative Sample (یعنی آن Anchor شامل Object یعنی نیست و Background است) و در صورت یک بودن به معنای Positive Sample (یعنی آن Anchor شامل Object است و Foreground است) می‌باشد. نحوه بدست آمدن آن نیز بر معنای شاخص Intersection-over-Union(IoU) می‌باشد. در صورتی که آن Anchor با هر یک از Ground-truth-box‌ها (منظور External Rectangular‌ها) داشته باشد و مقدار IoU overlap با اتر از 0.7 داشته باشد، آنگاه Object‌های مشخص شده در عکس‌های Training set می‌باشد، در صورتی که با همه Ground-truth-box‌ها IoU overlap از 0.3 داشته باشد نیز Negative sample می‌باشد و P برابر ۰ می‌شود. (Anchor هایی که در دو دسته بالا قرار نگیرند Invalid هستند و در تمرین استفاده نمی‌شوند)

بخش اول رابطه برای تعریف هزینه بخش Objectness Score آن Box را تعیین می‌کند استفاده می‌شود. تابع F_{cls} یک Cross entropy loss function می‌باشد. همان‌ P_i^* همان Label برای Anchor هاست که مطابق پاراگراف قبل تعیین می‌شود. P_i نیز خروجی Classification برای آن هاست که نشان‌دهنده احتمال وجود یک Object در آن Anchor می‌باشد. هدف از تعریف این بخش تابع این است که با کاهش Loss، پیش‌بینی اینکه Object شامل Box هست یا خیر را دقیق‌تر کند و به P_i^* برساند.

بخش دوم برای تعریف هزینه بخش Regression که مقادیر Offset آن Box را تعیین می‌کند استفاده می‌شود. تابع هزینه F_{reg} یک Smooth L1 Loss می‌باشد. این ترم در P_i^* ضرب شده؛ پس تنها در صورتی

این Loss اعمال می‌شود که یک Positive Sample باشد و واقعاً Object بوده باشد که منطقی نیز هست. δ بردار ۶ تایی هست که برابر با Offset پیش‌بینی شده توسط شاخه Regression می‌باشد. از طرفی t^* بردار آفست هدف برای a می‌باشد که نشان دهنده تبدیل ایده آل از Anchor Box فعلی به Ground-truth box هدف است. برای محاسبه آن نیز از پارامترهای Anchor Box و همچنین پارامترهای باکس خارجی افقی برای Ground-truth oriented box مطابق رابطه ۴ مقاله استفاده می‌کنیم.

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

شکل ۴ مقاله نیز نحوه مشخص شدن این پارامترها برای یک Box را نمایش داده است.

به طور خلاصه، ترم دوم با مشخص کردن Offset مورد نیاز برای تبدیل Anchor Box فعلی به باکس هدف، سعی می‌کند با کاهش Loss، بردار Offset پیش‌بینی شده توسط RPN را به این مقدار نزدیک‌تر کند (البته در صورتی که Object شامل Anchor Box باشد).

Rotated RoI Align

الف: نحوه عملکرد

هدف استفاده از Rotated RoIAlign، استخراج ویژگی‌ها با سایز ثابت از هر یک از Oriented Proposal ها- مستقل از زاویه چرخش آن- می‌باشد. از این بردار ویژگی در مرحله بعد برای Classification و استفاده می‌شود. استفاده از این عملیات باعث استخراج ویژگی بهتر برای اشیا با زوایای متفاوت می‌شود.

نحوه عملکرد آن را در ادامه توضیح می‌دهیم:

Oriented proposal به عنوان ورودی FPN های Feature map و همچنین FPN های Rotated RoIAlign را می‌گیرد. Proposal های تولید شده توسط RPN (که با ۶ پارامتر در نمایش Midpoint offset مشخص می‌شوند) عموماً به شکل متوازی‌الاضلاع هستند. با استفاده از رابطه ۲ مقاله می‌توان از روی پارامترهای باکس، مختصات ۴ راس این متوازی‌الاضلاع را بدست آورد.

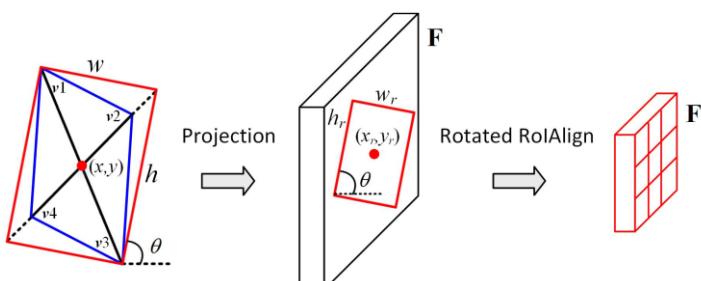
در مرحله اول برای ساده تر شدن محاسبات در ادامه کار، این متوازی‌الاضلاع را به یک باکس مستطیلی تبدیل می‌کنیم. برای این کار به سادگی می‌توان قطر کوچک متوازی‌الاضلاع را بزرگ‌تر نمود تا به اندازه قطر بزرگ برسد.

در مرحله دوم، نمایش باکس مستطیلی را به صورت $(x.w.y.h.\theta)$ در می‌آوریم که به ترتیب مختصات مرکز مستطیل، طول و عرض آن و زاویه ضلع بلند مستطیل با محور x را نشان میدهد. این تبدیل نیز به سادگی و از روی مختصات رئوس مستطیل انجام می‌شود.

در مرحله سوم، این باکس مستطیلی را روی Feature map متناظر با آن (که Proposal از آن استخراج شده است) نگاشت می‌کنیم. برای اینکار طبق رابطه ۵ مقاله، ابعاد را بر Stride آن Feature map تقسیم Downsampling نشان دهنده Stride (Feature map برویم). می‌کنیم تا از فضای تصویر اصلی به فضای feature map نگاشت آن factor آن feature map نسبت به تصویر اصلی می‌باشد). زاویه نیز در این نگاشت منطقاً تغییری نمیکند. حال هر ROI بحسب آمده را به گردید ۷ در ۷ تقسیم کرده و برای هر خانه گردید، از Feature map نمونه برداری می‌کنیم. با این کار سایز feature map خروجی را برای همه Proposal‌ها یکسان می‌نماییم. مقدار یک خانه گردید (در هر کanal) طبق رابطه زیر حساب می‌شود:

$$\mathbf{F}'_c(i, j) = \sum_{(x,y) \in \text{area}(i,j)} \mathbf{F}_c(R(x, y, \theta)) / n$$

همانطور که می‌بینیم باید میانگینی از سمپل‌های Feature map که در آن خانه قرار دارند بگیریم. ولی برای اینکه Sampling grid را با زاویه ROI بحسب آمده همسو کنیم، ($R(\cdot)$) Rotation Transformation را به سمپل‌ها اعمال می‌کنیم. با این کار فارغ از زاویه Object، فیچرهای سمپل برداری شده مطابق با Oriented Proposal می‌باشد و نمونه برداری دقیق تری انجام می‌شود.



شکل ۱۱ : مراحل Rotated RoIAlign

شکل بالا که در مقاله آورده شده، این مراحل را نشان میدهد. تصویر اول مراحل ۱ و ۲ را نشان میدهد که Proposal را به شکل مستطیلی در آورده و به نمایش ۵ پارامتری (با تعیین زاویه) تبدیل نموده است. تصویر دوم مرحله سوم را نشان میدهد که باکس مستطیلی را روی Feature map متناظر با آن نگاشت کرده است. سپس این مستطیل را به گردید ۷ در ۷ تقسیم کرده و با رابطه نشان داده شده در بالا (با اعمال $R(\cdot)$ با میانگین گیری از سمپل‌های مپ F ، گردید های متناظر مپ F' را می‌سازد).

ب: مشکلات عدم استفاده

در صورت عدم استفاده از Rotated RoIAlign و برای مثال استفاده از Horizontal RoI، نمی‌توانیم موقعیت دقیق Oriented Object را تشخیص بدهیم و زاویه آن قاعده‌تا با زاویه Object همسو نیست. به همین جهت بخشی از Box بی که در Feature map برای نمونه برداری مشخص می‌کنیم را بک‌گراند تصویر و یا اشیا دیگر تشکیل می‌دهند. حال اگر ما از این مپ نمونه برداری کرده تا Feature map خروجی که شامل ویژگی‌های استخراج شده هست را تشکیل دهیم، همانطور که می‌بینیم این ویژگی‌ها علاوه بر شی مورد نظر، شامل ویژگی‌های بک‌گراند و یا اشیا دیگر در نزدیکی شی مورد نظرمان نیز هستند که ماهیت تصادفی دارند و جزو ویژگی‌های اصلی محسوب نمی‌شوند. به همین دلیل عملکرد مدل در یادگیری ویژگی‌های Object ها مختل می‌شود و طبیعتاً انتظار می‌رود عملکرد ضعیفتری نشان دهد.



شکل ۱۲ : استفاده از Horizontal ROI

تصویر بالا نمونه‌ای از تصویر ورودی شبکه را نشان میدهد. همانطور که می‌بینیم در صورت استفاده از Horizontal ROI، مستطیلی که شامل جسم با کلاس نارنجی می‌شود، بخش زیادی از یک جسم دیگر با کلاس بنفس را نیز شامل می‌شود. بنابراین ویژگی‌های استخراج شده از این Proposal، شامل هردوی این کلاس‌ها می‌شود و حتی امکان دارد شبکه تشخیص اشتباہی روی کلاس آن بدهد.

عملکرد و کارایی

الف: چگونگی مستabil به دقت و کارایی بالا

شبکه Oriented R-CNN ارائه شده با برطرف کردن مشکلات شبکه‌های ۲ مرحله‌ای Oriented proposal (مخصوصاً در تولید Object Detection) توانست به عملکرد قابل توجهی در مقایسه با آن‌ها برسد.

در استیج اول با استفاده از Oriented RPN ارائه شده (که از نمایش Midpoint offset بهره می‌برد) می‌تواند در تشخیص oriented proposal ها بسیار خوب عمل کند. در جدول ۲ مقاله، شاخص Recall این بخش هنگام استفاده از 2000 Proposal 92.8% بdest آمده است. (البته برای ۱۰۰۰ پروپوزال نیز افت دقت ناچیزی را متحمل می‌شود که می‌تواند انتخاب مناسب تری باشد). همچنین تصویر ۶ مقاله، پروپوزال

های بدست آمده را روی تصاویر ورودی مختلف رسم کرده که نشان میدهد به خوبی اجسام مختلف را تشخیص داده است.

جدول ۲ مقاله نتایج مقایسه مدل ارائه شده با مدل های قبلی را بر روی دیتابست DOTA نشان میدهد. مطابق این نتایج مدل ارائه شده با بکبن R-50-FPN به دقت ۸۰.۸۷٪ میرسد که بیش از ۵ درصد از تمام مدل های ۱ استیچ و ۲ استیچی سابق عملکرد بهتری دارد.

جدول ۳ مقاله نیز این مقایسه را بر روی دیتابست HRSC2016 انجام داده که باز هم بهترین دقت بین شبکه ها نشان داده است.

جدول ۴ مقاله، سرعت اجرای مدل ها را مقایسه نموده است. مدل پیشنهادی به سرعت پردازش ۱۵.۱fps با بالاترین دقت میان شبکه های مقایسه شده رسیده است و عملکرد بهتری-هم از نظر دقت و هم سرعت- از مدل های ۲ استیچی داشته است. (تنها یک مدل ۱ استیچی سرعت ۱ fps بالاتری نشان داده که دقت آن حدود ۷ درصد پایینتر از مدل ارائه شده است)

تصویر ۹ مقاله هم سرعت در مقایسه با دقت را برای چندین مدل نشان داده که باز هم عملکرد خوب مدل Oriented R-CNN در هردو را نشان میدهد.

ب: عوامل موثر در کارایی محاسباتی مدل

به دلیل ساختار ساده مدل در بخش RPN (که از نمایش Midpoint offset ویژگی های روش های قبلی (مانند استفاده از RoI و Rotated Anchor box و یا transformer که تعداد پارامترهای به مراتب بالاتری از مدل ارائه شده دارند)، بهینه تر و با هزینه و وقت کمتری انجام میشود؛ در عین حال که خروجی های خوبی نیز تولید میکند. برای همین کارایی مدل به مراتب افزایش می یابد.

همچنین استفاده از Rotated RoIAlign نیز باعث استخراج بهتر ویژگی های اجسام نسبت به متعدد های قبلی مانند Horizontal ROI شده است که باعث بهبود دقت مدل می شود.

ج: مقایسه با سایر آشکارساز های ۲ مرحله ای

مطابق جدول ۲ مقاله، دقت این مدل از سایر آشکارساز های ۲ مرحله ای مقایسه شده بهتر است. مهمترین نقطه قوت این مدل همانطور که قبلا نیز گفته شده، توانایی آن در تولید مستقیم و بهینه Convolution Proposal ها می باشد. بخش Oriented RPN تنها از چندین لایه سبک Tشكيل ROI transformer است. بنابراین در مقایسه با مدلی که از Rotated Anchor استفاده میکند و یا تعداد پارامترهای به مراتب کمتر و دقت بهتری دارد.

جدول ۴ نیز دقت در مقایسه با سرعت را نشان میدهد که میبینیم از مدل های ۲ مرحله ای بالا در هر دو شاخص عملکرد بهتری دارد.

بخش دوم: پیاده‌سازی عملی

راهاندازی محیط و آماده‌سازی مجموعه داده

ابتدا مجموعه داده HRSC2016 دانلود نموده و به پیش‌پردازش آن میپردازیم.

این دیتا ست در پوشه AllImages تمام تصاویر را ذخیره کرده و باکس‌های متناظر با هر تصویر را در پوشه Annotations با فرمت xml. ذخیره کرده است.

تصویر زیر ساختار یکی از این فایل‌ها را نشان می‌دهد.

```
<annotation>
  <copyright>Aurora Group, VIP Lab, Xidian University</copyright>
  <contributors>Weiming Chen, Zizheng Ren, Bing Han, Zheng Yang, Yang Zhou, Xiaoyue Huang</contributors>
  <dataset>HRSC2016-MS</dataset>
  <source_dataset>HRSC2016</source_dataset>
  <label_level>L1</label_level>
  <filename>100000003</filename>
  <size>
    <width>1155</width>
    <height>820</height>
    <depth>3</depth>
  </size>
  <object>
    <type>bndbox</type>
    <name>ship</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>277</xmin>
      <ymin>326</ymin>
      <xmax>987</xmax>
      <ymax>698</ymax>
    </bndbox>
    <robndbox>
      <cx>644.5836</cx>
      <cy>521.053</cy>
      <w>168.8657</w>
      <h>731.003</h>
      <angle>1.941593</angle>
    </robndbox>
  </object>
  <object>
    <type>bndbox</type>
    <name>ship</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>322</xmin>
      <ymin>296</ymin>
      <xmax>458</xmax>
      <ymax>363</ymax>
    </bndbox>
    <robndbox>
      <cx>390.5168</cx>
      <cy>326.916</cy>
      <w>29.9405</w>
      <h>135.6597</h>
      <angle>1.88165</angle>
    </robndbox>
  </object>
</annotation>
```

شکل ۱۳ : نمونه ای از یک فایل Annotation

در ابتدای این فایل مشخصات کلی (مانند نام فایل تصویر، ابعاد آن و ...) ذکر شده است. سپس اطلاعات هر شی موجود در تصویر را با تگ object مشخص نموده است. ابتدای مشخصات آن شی (مانند کلاس آن و یا دشواری تشخیص شی) را مشخص نموده و سپس ۲ باکس مختلف برای نمایش هر شی مشخص نموده است. باکس اول با تگ bndbox مشخص شده و نشان دهنده یک باکس افقی معمول در اطراف شی است. باکس دوم با تگ robndbox، باکس چرخشی (Oriented) که برای تعیین دقیق موقعیت جسم استفاده میشود را تعیین می‌کند. این باکس را با پنج پارامتر مختصات x, y مرکز باکس، طول و عرض باکس چرخشی و زاویه چرخش باکس مشخص می‌کند.

به تعداد اشیا داخل تصویر، این تگ‌های object تکرار می‌شوند (در این نمونه ۲ شی در تصویر داریم). همانطور که در بخش تئوری مشاهده نمودیم، در این پیاده سازی میخواهیم نمایش معمول باکس‌های چرخشی با ۵ پارامتر (مانند همین فایل) را به نمایش midpoint offset با ۶ پارامتر تبدیل کنیم.

برای این کار کلاس HRSC2016Dataset را تعریف نموده تا داده‌ها را بارگذاری کرده و باکس‌ها را به نمایش مد نظر تبدیل کند.

برای هر نمونه تصویر، ابتدا فایل تصویر و Annotation متناظرش را میخوانیم و با فایل، آن را میخوانیم و ۵ پارامتر توصیف کننده باکس را استخراج میکنیم.

تصویر زیر این مراحل را نشان می‌دهد:

```
def __getitem__(self, idx):
    img_id = self.image_ids[idx]
    img_path = os.path.join(self.image_dir, f"{img_id}.bmp")
    ann_path = os.path.join(self.annotation_dir, f"{img_id}.xml")

    image = Image.open(img_path).convert("RGB")

    tree = ET.parse(ann_path)
    root = tree.getroot()

    boxes = []
    labels = []

    for obj in root.findall("object"):
        robndbox = obj.find("robndbox")
        if robndbox is None:
            continue

        cx = float(robndbox.find("cx").text)
        cy = float(robndbox.find("cy").text)
        w = float(robndbox.find("w").text)
        h = float(robndbox.find("h").text)
        angle = float(robndbox.find("angle").text)
```

شکل ۱۴ : بارگذاری داده‌های یک نمونه تصویر

حال باید تبدیل به نمایش Midpoint offset را انجام دهیم، مختصات مرکز تصویر در نمایش midpoint offset تفاوتی با نمایش فعلی ندارد و همان نقطه مرکز باکس میباشد.

برای محاسبه پارامتر های دیگر در مرحله اول مختصات 4 راس باکس چرخش یافته را محاسبه میکنیم.
برای این کار از فرمول دوران به اندازه یک زاویه حول نقطه (x, y) که مطابق زیر است استفاده میکنیم:

$$\begin{cases} x' = c_x + \sin(\theta) * shifted_y - \cos(\theta) * shifted_x \\ y' = c_y + \cos(\theta) * shifted_y + \sin(\theta) * shifted_x \end{cases}$$

با استفاده از فرمول بالا، باکس با مرکز (c_x, c_y) را حول همین نقطه به اندازه θ دوران میدهیم که
مختصات رئوس به صورت زیر میشود.

```
v1 = (cx - dx * cos_a + dy * sin_a, cy - dx * sin_a - dy * cos_a) # top-left
v2 = (cx + dx * cos_a + dy * sin_a, cy + dx * sin_a - dy * cos_a) # top-right
v3 = (cx + dx * cos_a - dy * sin_a, cy + dx * sin_a + dy * cos_a) # bottom-right
v4 = (cx - dx * cos_a - dy * sin_a, cy - dx * sin_a + dy * cos_a) # bottom-left
```

شکل ۱۵ : مختصات رئوس باکس چرخش یافته

حال طول و عرض باکس خارجی را در نمایش Midpoint offset محاسبه میکنیم. برای اینکار به سادگی
میتوانیم فاصله بین بزرگترین X رئوس و کوچکترین آن ها را محاسبه کنیم که طول باکس خارجی، و
فاصله بین بزرگترین Y رئوس و کوچکترین آن ها را محاسبه کنیم که ارتفاع باکس خارجی میشود.

برای محاسبه مقادیر آفست $\Delta\alpha, \Delta\beta$ باید مختصات نقاط میانی ضلع بالایی و ضلع راستی باکس
خارجی را به دست آوریم که به سادگی و با کمک مختصات رئوس و طول و ارتفاع باکس خارجی بدست
میآیند. حال بسته به اینکه زاویه باکس بزرگتر یا کوچکتر از صفر باشد دو حالت میتوانند پیش بیايد. در
حالتی که زاویه مثبت باشد، راس $V1$ همان راس بالا سمت راست میباشد و بقیه به صورت ساعتگرد تعیین
میشوند. ولی در صورتی که زاویه منفی باشد راس $V1$ در واقع راس پایین راست میشود و بقیه رئوس بعد
از آن به ترتیب ساعتگرد مشخص میشوند. این موضوع در تعیین مقادیر افست اهمیت دارد.

در حالت اول، برای تعیین افست $\Delta\alpha$ میباشد فاصله X راس $V1$ با نقطه میانی بالا را محاسبه و برای
بدست اوردن افست $\Delta\beta$ فاصله Y راس $V2$ با نقطه میانی راست را حساب کنیم.

ولی در حالت دوم، برای تعیین افست $\Delta\alpha$ میباشد فاصله X راس $V4$ با نقطه میانی بالا را محاسبه و
برای بدست اوردن افست $\Delta\beta$ فاصله Y راس $V1$ با نقطه میانی راست را حساب کنیم.

برای تعیین این حالت ها نیز به سادگی میتوان مقدار X رئوس $V1$ و $V2$ را مقایسه کنیم.

```

top_mid = (x, y - h_ext / 2)
right_mid = (x + w_ext / 2, y)

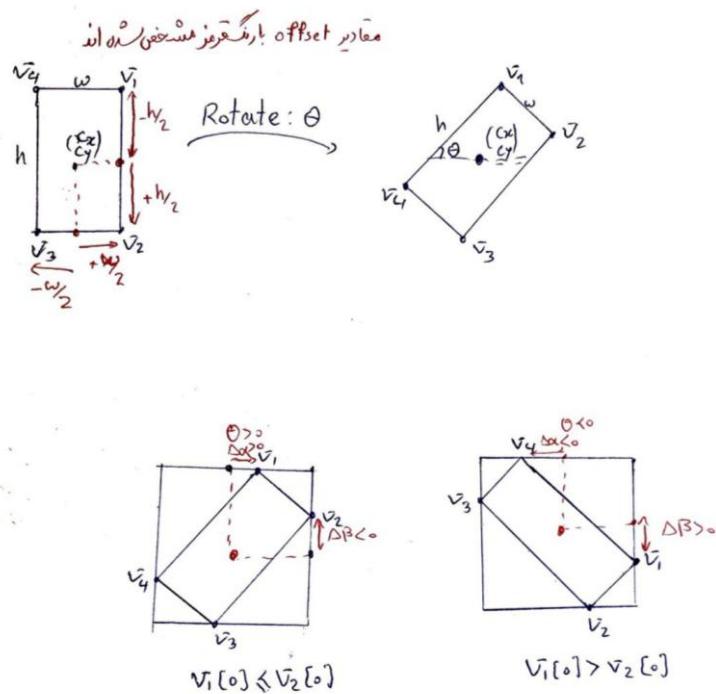
if v1[0] <= v2[0]:
    delta_alpha = v1[0] - top_mid[0]
    delta_beta = v2[1] - right_mid[1]
else:
    delta_alpha = v4[0] - top_mid[0]
    delta_beta = v1[1] - right_mid[1]

boxes.append([x, y, w_ext, h_ext, delta_alpha, delta_beta])
labels.append(1)

```

شکل ۱۶ : تعیین مقادیر آفست

حال با مشخص شدن همه پارامترها برای نمایش Midpoint offset باکس، آن را به یک لیست اضافه کرده و این کار را برای باکس های دیگر تصویر نیز در صورت وجود انجام میدهیم.



شکل ۱۷ : نحوه چرخش و بدست آوردن مختصات رئوس و مقادیر آفست

تصویر بالا دید بهتری از نحوه اعمال چرخش بر روی باکس و بدست آوردن مختصات رئوس می‌دهد.

حال برای رسم باکس ها ببروی تصویر، یکتابع مینویسیم که باکس های هر تصویر را ببروی آن رسم بکند. برای رسم نیز نیاز است مطابق فرمول ۲ مقاله، نمایش Midpoint offset را با تبدیل ساده ای به مختصات رئوس مستطیل باکس تبدیل کرده تا بتوان آن را رسم نمود.

```

def draw_midpoint_offset_box(ax, box, color='red'):
    x, y, w, h, da, db = box.tolist()

    v1 = (x + da, y - h/2)
    v2 = (x + w/2, y + db)
    v3 = (x - da, y + h/2)
    v4 = (x - w/2, y - db)

    polygon = patches.Polygon([v1, v2, v3, v4], closed=True, fill=False, edgecolor=color, linewidth=2)
    ax.add_patch(polygon)

def visualize_examples(dataset, num_samples=3):
    for i in range(num_samples):
        image, target = dataset[i]
        image_np = image.permute(1, 2, 0).numpy()

        fig, ax = plt.subplots(1, 1, figsize=(8, 8))
        ax.imshow(image_np)

        for box in target['boxes']:
            draw_midpoint_offset_box(ax, box)

        ax.set_title(f"Sample {i} with {len(target['boxes'])} bounding box(es)")
        plt.axis('off')
        plt.show()

```

شکل ۱۸ : نحوه تبدیل نمایش Midpoint offset برای رسم باکس ها

در نهایت نتیجه را برای تعدادی از تصاویر رسم می کنیم:

Sample 2 with 5 bounding box(es)



شکل ۱۹ : بارگذاری نمونه اول

Sample 3 with 9 bounding box(es)



شکل ۲۰ : بارگذاری نمونه دوم

Sample 4 with 5 bounding box(es)



شکل ۲۱ : بارگذاری نمونه سوم

همانطور که میبینیم تبدیل نمایش باکس ها به درستی صورت گرفته و بعد از رسم، باکس ها کاملا بروی کشتی ها محاط شده اند.

آموزش مدل Oriented R-CNN

حال مدل را مطابق مقاله تعریف میکنیم تا ببروی دیتا ست داده شده آموزش بدھیم.