

به نام خدا



دانشگاه تهران



دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

نام عضو اول	فاطمه رشیدی شهری
شماره دانشجویی	۶۱۰۳۹۹۱۳۱
نام عضو دوم	آراد وزیرپناه
شماره دانشجویی	۶۱۰۳۹۹۱۸۲

فهرست

1	قوانین
1	پرسش 1. پیاده‌سازی مدل U-Net
1	1-1. آماده سازی مجموعه داده
2	1-2. پیاده سازی مدل
4	1-3. تقویت داده
6	1-4. بهینه‌ساز، متریک‌ها و تابع هزینه
8	1-5. آموزش مدل
10	1-6. ارزیابی مدل
1	پرسش 2 - تشخیص موجودات زیر آب
1	1-2. معرفی مدل Faster R-CNN
2	2-2. سوالات تشریحی
2	2-2-1. مقایسه‌ی مدل‌های region-based CNNs
4	2-2-2. مقایسه‌ی مدل‌های one-stage و two-stage
8	2-2-3. GIOU, Soft-NMS, OHEM
11	3-2. معرفی مجموعه دادگان
11	3-2-1. EDA و پیش پردازش دادگان
14	3-2-2. تقویت داده
16	3-2-3. ساخت دیتالودر
18	4-2. تعریف مسئله
18	4-2-1. طراحی معماری Faster R-CNN
22	4-2-2. طراحی Region Proposal Network
27	4-2-3. آموزش مدل

شکل‌ها

1. شکل 1. نمایش تعدادی از تصاویر به همراه maskهای متناظر با آنها در مجموعه‌ی دادگان.....
2. شکل 2. معماری مدل U-Net پیاده شده
3. شکل 3. تعداد پارامترها برای مدل U-Net با ساختار گفته شده
4. شکل 4. تعدادی از تصاویر آموزشی به همراه maskهای متناظرشان پس از اعمال augmentation.....
5. شکل 5. Confusion matrix.....
6. شکل 6. شکل متناظر با فرمول با محاسبه‌ی Dice coefficient
7. شکل 7. فرمول متناظر با محاسبه‌ی IoU Score
8. شکل 8. نمودار متناظر با Dice، IoU، Loss و Accuracy برای دو مجموعه‌ی تصاویر آموزشی و اعتبارسنجی حین یادگیری مدل
9. شکل 9. Maskهای اصلی و پیش‌بینی شده توسط شبکه برای تعدادی از داده‌های ارزیابی به همراه تصاویر اصل
10. شکل 10. معماری شبکه‌ی RCNN.....
11. شکل 11. معماری شبکه‌ی Fast RCNN
12. شکل 12. معماری شبکه‌ی Faster RCNN
13. شکل 13. نحوه‌ی کارکرد تکنیک OHEM برای مسئله‌ی تشخیص اشیا در تصویر
14. شکل 14. نمودار مربوط به فراوانی موجودات در هر تصویر
15. شکل 15. نمودار مربوط به فراوانی کلاس‌ها در هر تصویر
16. شکل 16. توزیع انواع موجودات در کل مجموعه‌ها
17. شکل 17. نمودار مربوط به توزیع تعداد تصاویر شامل هر کلاس
18. شکل 18. نمونه‌هایی از تقویت داده
19. شکل 19. نمای کلی Faster RCNN
20. شکل 20. Feature Mapهای بدست آمده از ResNet101.....
21. شکل 21. توزیع anchor points
22. شکل 22. توزیع سائز جعبه‌های مرزی و نقاط مرکزی به دست آمده
23. شکل 23. Anchor boxهای روی یکی از تصاویر
24. شکل 24. Anchor boxes برای یک anchor point

جدول‌ها

جدول 1. تعداد تصاویر در سه مجموعه‌ی آموزشی، اعتبارسنجی و ارزیابی پس از تقسیم دادگان.....1

جدول 2. نتایج ارزیابی شبکه‌ی آموزش دیده روی داده‌های ارزیابی10

قبل از پاسخ دادن به پرسش‌ها، موارد زیر را با دقت مطالعه نمایید:

- از پاسخ‌های خود یک گزارش در قالبی که در صفحه‌ی درس در سامانه‌ی Elearn با نام **REPORTS_TEMPLATE.docx** قرار داده شده تهیه نمایید.
- پیشنهاد می‌شود تمرین‌ها را در قالب گروه‌های دو نفره انجام دهید. (بیش از دو نفر مجاز نیست و تحویل تک نفره نیز نمره‌ی اضافی ندارد) توجه نمایید الزامی در یکسان ماندن اعضای گروه تا انتهای ترم وجود ندارد. (یعنی، می‌توانید تمرین اول را با شخص A و تمرین دوم را با شخص B و ... انجام دهید)
- **کیفیت گزارش شما در فرآیند تصحیح از اهمیت ویژه‌ای برخوردار است؛** بنابراین، لطفاً تمامی نکات و فرض‌هایی را که در پیاده‌سازی‌ها و محاسبات خود در نظر می‌گیرید در گزارش ذکر کنید.
- در گزارش خود مطابق با آنچه در قالب نمونه قرار داده شده، برای شکل‌ها زیرنویس و برای جدول‌ها بالانویس در نظر بگیرید.
- الزامی به ارائه توضیح جزئیات کد در گزارش نیست، اما باید نتایج بدست آمده از آن را گزارش و تحلیل کنید.
- **تحلیل نتایج الزامی می‌باشد، حتی اگر در صورت پرسش اشاره‌ای به آن نشده باشد.**
- **دستیاران آموزشی ملزم به اجرا کردن کدهای شما نیستند؛** بنابراین، هرگونه نتیجه و یا تحلیلی که در صورت پرسش از شما خواسته شده را به طور واضح و کامل در گزارش بیاورید. در صورت عدم رعایت این مورد، بدیهی است که از نمره تمرین کسر می‌شود.
- **کدها حتماً باید در قالب نوت‌بوک با پسوند ipynb تهیه شوند، در پایان کار، تمامی کد اجرا شود و خروجی هر سلول حتماً در این فایل ارسالی شما ذخیره شده باشد.** بنابراین برای مثال اگر خروجی سلولی یک نمودار است که در گزارش آورده‌اید، این نمودار باید هم در گزارش هم در نوت‌بوک کدها وجود داشته باشد.
- **در صورت مشاهده‌ی تقلب امتیاز تمامی افراد شرکت‌کننده در آن، 100- لحاظ می‌شود.**
- تنها زبان برنامه نویسی مجاز **Python** است.
- استفاده از کدهای آماده برای تمرین‌ها به هیچ وجه مجاز نیست. در صورتی که دو گروه از یک منبع مشترک استفاده کنند و کدهای مشابه تحویل دهند، تقلب محسوب می‌شود.

- نحوه محاسبه تاخیر به این شکل است: پس از پایان رسیدن مهلت ارسال گزارش، حداکثر تا یک هفته امکان ارسال با تاخیر وجود دارد، پس از این یک هفته نمره آن تکلیف برای شما صفر خواهد شد.

○ سه روز اول: بدون جریمه

○ روز چهارم: ۵ درصد

○ روز پنجم: ۱۰ درصد

○ روز ششم: ۱۵ درصد

○ روز هفتم: ۲۰ درصد

- حداکثر نمره‌ای که برای هر سوال می‌توان اخذ کرد ۱۰۰ بوده و اگر مجموع بارم یک سوال بیشتر از ۱۰۰ باشد، در صورت اخذ نمره بیشتر از ۱۰۰، اعمال نخواهد شد.

○ برای مثال: اگر نمره اخذ شده از سوال ۱ برابر ۱۰۵ و نمره سوال ۲ برابر ۹۵ باشد، نمره نهایی تمرین ۹۷.۵ خواهد بود و نه ۱۰۰.

- لطفا گزارش، کدها و سایر ضمایم را به در یک پوشه با نام زیر قرار داده و آن را فشرده سازید، سپس در سامانه‌ی Elearn بارگذاری نمایید:

HW[Number]_[Lastname]_[StudentNumber]_[Lastname]_[StudentNumber].zip

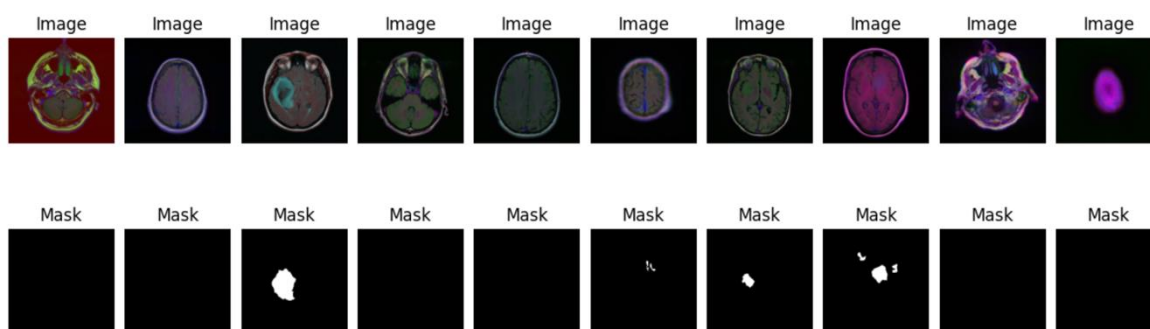
(مثال: HW1_Ahmadi_810199101_Bagheri_810199102.zip)

- برای گروه‌های دو نفره، بارگذاری تمرین از جانب یکی از اعضا کافی است ولی پیشنهاد می‌شود هر دو نفر بارگذاری نمایند.

پیش 1. پیاده سازی مدل U-Net

۱-۱. آماده سازی مجموعه داده

مجموعه‌ی دادگان داده‌شده شامل 110 مجموعه عکس از افراد مختلف به همراه mask برای هر تصویر است. برای دستیابی به تصاویر و ماسک‌های متناظر با آنها نیز لازم است هر دسته را به طور مجزا استخراج کرده و با عنوان‌های images و masks آنها را جدا کنیم. 10 مورد از این تصاویر و ماسک‌های متناظر با آنها را به تصادف انتخاب می‌کنیم و نمایش می‌دهیم:



شکل 1. نمایش تعدادی از تصاویر به همراه maskهای متناظر با آنها در مجموعه‌ی دادگان.

بعد از خواندن تصاویر، لازم است برای ادامه‌ی کار و آموزش مدل، تصاویر را به سه دسته‌ی آموزشی، اعتبارسنجی و ارزیابی تقسیم کنیم. برای این کار، 80% از تصاویر را به عنوان داده‌های آموزشی، 10% را به عنوان داده‌های اعتبارسنجی و 10% باقی مانده را نیز به عنوان داده‌های ارزیابی در نظر می‌گیریم. نهایتاً تعداد هر یک از دسته داده‌ها به صورت زیر خواهد بود:

Training Data	Validation Data	Test Data
3143	393	393

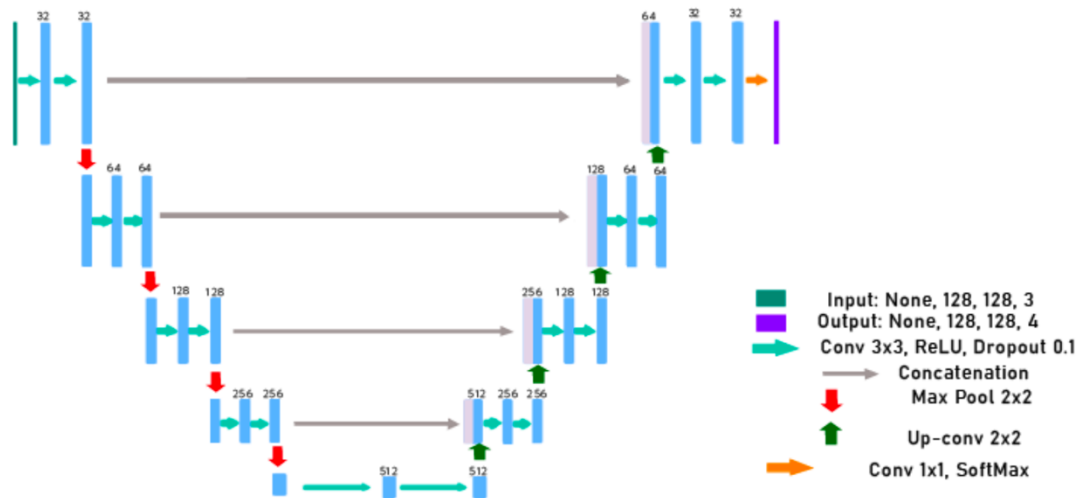
جدول 1. تعداد تصاویر در سه مجموعه‌ی آموزشی، اعتبارسنجی و ارزیابی پس از تقسیم دادگان

۱-۲. پیاده سازی مدل

در این بخش به پیاده سازی مدل U-Net را طبق معماری گفته شده در مقاله انجام میدهیم. همانطور که می‌دانیم، ساختار این مدل شامل دو بخش encoder و decoder است. فرم کلی آن به شرح زیر است:

- لایه‌ی ورودی با شکل تصاویر ورودی (128, 128, 3)
 - Encoder
 - دو لایه‌ی کانولوشنی با 32 فیلتر 3×3 و فعال‌ساز ReLU
 - لایه‌ی max-pooling با فیلتر 2×2
 - دو لایه‌ی کانولوشنی با 64 فیلتر 3×3 و فعال‌ساز ReLU
 - لایه‌ی max-pooling با فیلتر 2×2
 - دو لایه‌ی کانولوشنی با 128 فیلتر 3×3 و فعال‌ساز ReLU
 - لایه‌ی max-pooling با فیلتر 2×2
 - دو لایه‌ی کانولوشنی با 256 فیلتر 3×3 و فعال‌ساز ReLU
 - لایه‌ی max-pooling با فیلتر 2×2
 - Bottleneck
 - دو لایه‌ی کانولوشنی با 512 فیلتر 3×3 و فعال‌ساز ReLU
 - Decoder
 - یک لایه‌ی deconvolution با 256 فیلتر 2×2 و $\text{strides}=(2, 2)$
 - Concatenation با بخش متناظر در encoder
 - دو لایه‌ی کانولوشنی با 256 فیلتر 3×3 و فعال‌ساز ReLU
 - یک لایه‌ی deconvolution با 128 فیلتر 2×2 و $\text{strides}=(2, 2)$
 - Concatenation با بخش متناظر در encoder
 - دو لایه‌ی کانولوشنی با 128 فیلتر 3×3 و فعال‌ساز ReLU
 - یک لایه‌ی deconvolution با 64 فیلتر 2×2 و $\text{strides}=(2, 2)$
 - Concatenation با بخش متناظر در encoder
 - دو لایه‌ی کانولوشنی با 64 فیلتر 3×3 و فعال‌ساز ReLU
 - یک لایه‌ی deconvolution با 32 فیلتر 2×2 و $\text{strides}=(2, 2)$
 - Concatenation با بخش متناظر در encoder
 - دو لایه‌ی کانولوشنی با 32 فیلتر 3×3 و فعال‌ساز ReLU
 - لایه‌ی کانولوشنی خروجی با دو فیلتر 1×1 و فعال‌ساز softmax

شکل معماری توصیف شده در بالا، به صورت زیر است:



شکل 2. معماری مدل U-Net پیاده شده

همانطور که گفته شد، این مدل، دارای دو بخش اصلی است، encoder و decoder. در بخش اول، ویژگی‌های تصویر توسط لایه‌های کانولوشنی در سطوح مختلف استخراج می‌شوند و با استفاده از لایه‌های max-pooling، سایز feature map ها کاهش پیدا میکند. Bottleneck مثل یک پل بین دو بخش یاد شده عمل می‌کند و به مدل کمک می‌کند تا اطلاعات معنایی سطح بالا را ضبط کند. در بخش decoder مدل در هر مرحله ابتدا با استفاده از لایه‌های deconvolution ابعاد feature map ها را افزایش می‌دهد تا به ابعاد اولیه برسد و این map ها را با feature maps متناظر با آنها در بخش encoder با هم concatenate می‌کند. لایه‌های کانولوشنی در این بخش نیز منجر به بازتولید mask ها میشوند. در انتها نیز یک لایه خروجی داریم که کلاس هر پیکسل را پیش‌بینی میکند.

نحوه کار کلی این مدل به گونه‌ای است که در بخش اول رفته رفته با کاهش سایز feature map ها، receptive field را افزایش می‌دهد و با اینکار باعث می‌شود که شبکه به اطلاعات جزئی تصاویر دست پیدا کند. در بخش دوم نیز مدل با افزایش تدریجی سایز feature maps باعث می‌شود که شبکه، ویژگی‌ها را localize کند. Skip connection ها نیز با اتصال قسمت‌های مختلف مسیرهای encoding و decoding منجر به حفظ اطلاعات مهم و ضروری از لایه‌های قبلی می‌شوند و امکان تقسیم بندی دقیق تر را فراهم می‌کنند.

دلیل انتخاب این مدل برای انجام taskهای پزشکی نیز معماری آن است که ترکیبی از اطلاعات مکانی حاصل از مسیر downsampling و اطلاعات contextual حاصل از مسیر upsampling را ترکیب می‌کند تا segmentation map دقیقی را پیش‌بینی کند.

تعداد پارامترها برای مدل ساخته شده با ساختار فوق به صورت زیر است:

Total Params	Trainable Params	Non-trainable Params
7760130	7760130	0

شکل 3. تعداد پارامترها برای مدل **U-Net** با ساختار گفته شده

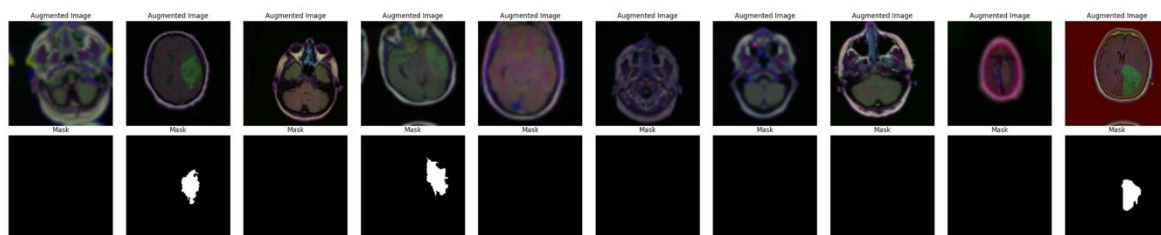
۳-۱. تقویت داده

برای تقویت داده‌ها از کتابخانه‌ی Albumentations استفاده کرده‌ایم. این کتابخانه شامل تعداد زیادی تبدیل است که با اعمال آنها روی تصاویر اصلی مجموعه دادگان، با هدف بهبود عملکرد شبکه‌های کانولوشنی عمیق، سعی در افزایش اندازه‌ی مجموعه دادگان دارد. با این کار، نیاز به label کردن دستی تصاویر کم‌تر می‌شود. از این رو، می‌توان تا حد زیادی از هزینه‌های تولید مجموعه دادگان کاست.

ما تبدیل‌های زیر را برای این منظور به صورت تصادفی روی تصاویر اعمال کرده‌ایم:

- **Resize**: اندازه‌ی هر یک از تصاویر و ماسک‌ها را طبق مقاله به سایز (128, 128) تغییر می‌دهیم. این کار منجر به سازگاری بیشتر دادگان با مدل می‌شود. همچنین، محاسبات را بهینه می‌کند.
- **Rotate**: چرخش با احتمال 50% و در بازه‌ی 15- تا 15 درجه. این کار منجر می‌شود روی حدود نیمی از تصاویر آموزشی چنین چرخشی اعمال شود. اهمیت این تبدیل نیز به این دلیل است که ممکن است تصاویر متفاوت در پوزیشن‌های مختلف از بیمار گرفته شده باشد یا اینکه بنا به تنظیمات دستگاه تصویربرداری، زاویه‌ی تمام عکس‌ها یکسان نباشند. با این کار، **generality** و **robustness** مدل بیشتر می‌شود.
- **RandomResizedCrop**: با اعمال این تبدیل روی تصاویر، برش‌های مختلفی از تصاویر با اندازه‌ی (128, 128) به صورت تصادفی (با احتمال 50%) ایجاد می‌شود. این کار باعث می‌شود مدل تصاویر را در مقیاس‌های متفاوت ببیند و ویژگی‌های مورد نیاز را بهتر استخراج کند. در واقع، با برش تصاویر، روی بخش‌های باقی مانده تمرکز بیشتری می‌کند و جزئیات را بهتر یاد می‌گیرد.

- ShiftScaleRotate: این تبدیل، ترکیبی از سه تبدیل shift، scale و rotation است. چون در تقویت داده به صورت مجزا از تبدیل چرخش استفاده میکنیم، در این تبدیل، rotate limit برابر صفر قرار می‌دهیم تا چرخشی را اعمال نکند. shift دادن و scale کردن تصاویر طبق این تصویر منجر می‌شود تغییراتی جزئی در پوزیشن و اندازه‌ی تصویر اعمال شود که ممکن است به علت نحوه‌ی قرار گیری بیمار زیر دستگاه تصویر برداری و یا تنظیمات دستگاه رخ دهد. واضح است که با این کار نیز عملکرد مدل بهتر شده و general تر می‌شود.
- GaussianBlur: در هنگام تصویربرداری ممکن است در اثر فوکوس نامناسب، نزدیکی بیش از حد دوربین به بافت و یا تکان خوردن بیمار یا دستگاه، تصویر اندکی تار شود. با اعمال این تبدیل به طور تصادفی روی حدود 50% داده‌های آموزشی، تصاویر اندکی تار می‌شوند تا مدل یاد بگیرد بخش‌بندی را برای تصاویری که در شرایط غیر ایده‌آل ثبت شده‌اند نیز به درستی انجام دهد.



شکل 4. تعدادی از تصاویر آموزشی به همراه maskهای متناظرشان پس از اعمال augmentation

برای اعمال تبدیلات فوق روی داده‌های آموزشی، اعتبارسنجی و ارزیابی، یک datagenerator در نظر می‌گیریم و batch size را روی 16 تنظیم می‌کنیم. سپس، روی داده‌های آموزشی، اعمال تقویت داده را انجام می‌دهیم، اما روی داده‌های اعتبارسنجی و ارزیابی تنها تغییر سایز را اعمال می‌کنیم. به این صورت، تعدادی generator داریم که برای آموزش و ارزیابی مدل در ادامه از آنها استفاده خواهیم کرد.

تمام تبدیلهای فوق که به صورت تصادفی روی دادگان آموزشی اعمال می‌شوند، باعث می‌شوند مدل robustness بیشتری داشته باشد و تعدادی عکس جدید تولید میکند که مدل با دیدن آنها general تر میشود و عملکرد بهتری خواهد داشت.

۴-۱. بهینه‌ساز، متریک‌ها و تابع هزینه

با توجه به شکل زیر که نمایانگر confusion matrix است، به تعریف هر یک از متریک‌ها می‌پردازیم و کاربرد هر یک را به اختصار توضیح می‌دهیم.

		Ground truth	
		FTU (1)	Background (0)
Prediction	FTU (1)	TP	FP
	Background (0)	FN	TN

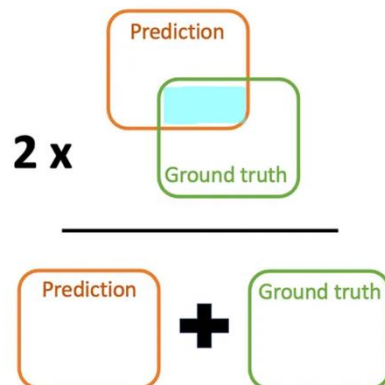
شکل ۵. Confusion matrix

اولین معیار، Dice coefficient یا همان f1-score است که کاربرد گسترده‌ای در ارزیابی مسائل segmentation پزشکی دارد و با استفاده از فرمول زیر محاسبه می‌شود:

$$\frac{2TP}{2TP + FP + FN}$$

معادله ۱. فرمول محاسبه‌ی **dice coefficient**

این معیار نسبتی از میزان اشتراک mask اصلی و پیش‌بینی شده توسط مدل، به مجموع این دو تصویر را گزارش می‌کند. در شکل زیر این مسئله واضح تر مطرح شده است:

$$\text{Dice} = \frac{2 \times \text{Area of overlap}}{\text{Total area}} = \frac{2 \times \text{Area of overlap}}{\text{Area of Prediction} + \text{Area of Ground truth}}$$


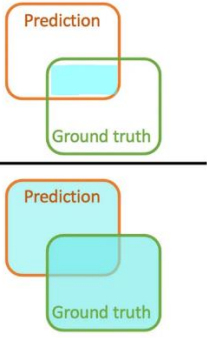
شکل ۶. شکل متناظر با فرمول با محاسبه‌ی **Dice coefficient**

درواقع این متریک نقاط مشترک بین دو mask اصلی و پیش‌بینی شده توسط مدل را حساب می‌کند و دو برابر آن را بر مجموع دو تصویر (و نه اجتماع) گزارش می‌کند. به این صورت، علاوه بر تاثیر دادن نقاطی که به درستی در score نهایی پیش‌بینی شده‌اند، نقاط false positive را نیز در نظر می‌گیرد. معیار دوم که معمولاً در این‌گونه مسائل گزارش می‌شود و با آن سر و کار داریم، IoU Score است. این معیار به Jaccard index نیز معروف است و با فرمول زیر محاسبه می‌شود:

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

معادله 2. فرمول محاسبه‌ی IoU Score

تفاوت این معیار با dice در این است که در صورت کسر تنها اشتراک دو mask پیش‌بینی شده و اصلی را داریم و در مخرج کسر نیز اجتماع پیکسل‌های دو تصویر را. شکل زیر نمایانگر نحوه‌ی محاسبه‌ی این معیار است:

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}} =$$


شکل 7. فرمول متناظر با محاسبه‌ی IoU Score

لازم به ذکر است که از آنجا که مسئله‌ای که در حال بررسی آن هستیم، باینری است (بافت تومور و بافت سالم)، میتوان از binary cross entropy به عنوان تابع هزینه برای آن استفاده کرد. همچنین، متریک دیگری که در ادامه آن را گزارش خواهیم کرد، accuracy است که از رابطه‌ی زیر بدست می‌آید:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

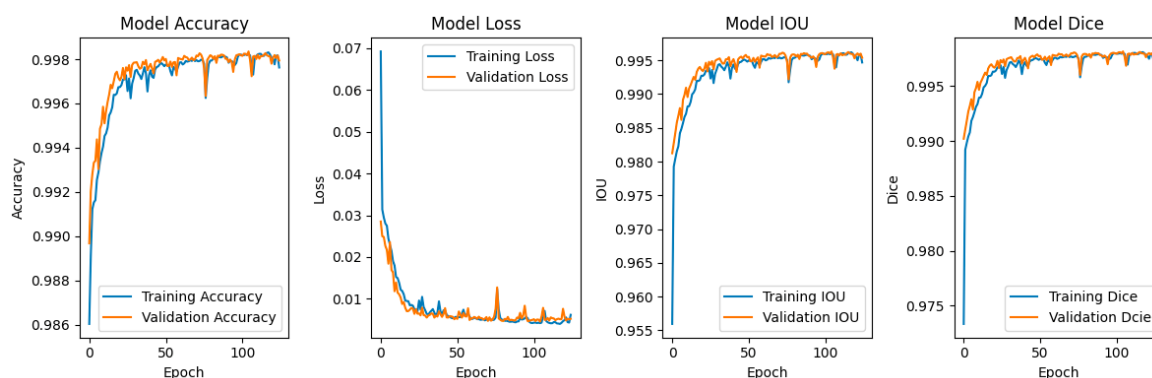
معادله 3. فرمول محاسبه‌ی Accuracy

در رابطه با بهینه ساز نیز همانطور که در مقاله گفته شده، از بهینه‌ساز adam با نرخ یادگیری 0.001 استفاده کرده‌ایم.

۵-۱. آموزش مدل

برای آموزش مدل ساخته شده، batch size را روی 16 تنظیم می‌کنیم. همچنین تعداد اپاک‌ها را نیز برابر 300 قرار می‌دهیم. اما در این میان از early stopping نیز استفاده می‌کنیم طوری‌که اگر اختلاف validation loss در 20 اپاک متوالی، تغییر چندانی نداشت، آموزش مدل متوقف شود. همچنین، بهترین وزن‌های شبکه را نیز ذخیره می‌کنیم تا در پایان گزارش کنیم در کدام اپاک مدل بهترین عملکرد را داشته است.

حال، به آموزش مدل می‌پردازیم. پس از بررسی نتایج، درمی‌یابیم که بعد از گذشت 125 اپاک، آموزش مدل با استفاده از early stopping متوقف می‌شود. همچنین، بنا به نتایج، بهترین وزن‌ها در اپاک 105 بدست آمده‌اند. در زیر نمودارهای متناظر با accuracy, loss, IoU و Dice را برای دو مجموعه‌ی تصاویر آموزشی و اعتبارسنجی مشاهده می‌کنیم:



شکل 8. نمودار متناظر با Accuracy و Loss IoU Dice برای دو مجموعه‌ی تصاویر آموزشی و اعتبارسنجی حین یادگیری مدل

در نمودار اول از سمت چپ، Accuracy نمایش داده شده است. همانطور که واضح است، با گذشت زمان و در حین یادگیری، این معیار سیر صعودی دارد. تقریباً در تمام مسیر یادگیری نیز accuracy برای دادگان اعتبارسنجی، بالاتر از دادگان آموزشی است. نزدیکی این دو نمودار به هم، به معنای این است که مدل تا حد خوبی generalize شده است و در تمام مسیر یادگیری روی داده‌های اعتبارسنجی نیز عملکرد خوب و رو به بهبودی دارد.

در نمودار بعدی، تغییرات loss را مشاهده می‌کنیم. واضح است که در ابتدا مقدار loss زیاد است اما با گذشت زمان، کاهش یافته و به صفر میل می‌کند. در واقع در روند آموزش مدل، خطای پیش‌بینی شبکه رفته رفته کاهش می‌یابد. با کنار هم قرار دادن این نمودار و نمودار accuracy، میتوان نتیجه گرفت از آنجا که با گذر زمان در آموزش مدل، accuracy پیوسته افزایش یافته و loss کم شده، پس شبکه عملکرد خوبی از خود ارائه داده است و یادگیری مطلوب است. اما از آنجا که بررسی دو معیار dice و IoU نیز برای این مسئله از اهمیت زیادی برخوردار است، لذا این دو نمودار را نیز در ادامه بررسی میکنیم.

شکل کلی نمودارهای متناظر با dice و IoU مشابه است. همچنین هر دو این نمودارها مانند نمودار accuracy سیر صعودی دارند و در ایپاک‌های نهایی نیز تقریباً ثابت می‌شوند. به بیان دیگر، از جایی به بعد، شبکه با پارامترهای تنظیم شده، ظرفیت بیشتری برای یادگیری ندارد. از آنجا که میزان این معیارها نیز در طی آموزش مدل رو به افزایش است و همچنین طبق تعریفشان، شبکه به خوبی در حال یادگیری است.

در تمام نمودارهای فوق، متریک‌ها برای هر دو مجموعه‌ی آموزشی و اعتبارسنجی خیلی به هم نزدیک اند. این نزدیکی بدین معناست که مدل دچار overfitting نشده است و روی داده‌های دیده نشده نیز عملکرد خوبی در هر ایپاک از خود نشان می‌دهد و در واقع این نزدیکی بیان می‌کند شبکه تصاویر را memorize نمیکند بلکه general است. همچنین، ثبات این نمودارها نشان می‌دهد که معماری مدل، پارامترها و فرایند یادگیری برای مسئله‌ی مورد نظر، به خوبی انتخاب و انجام شده‌اند.

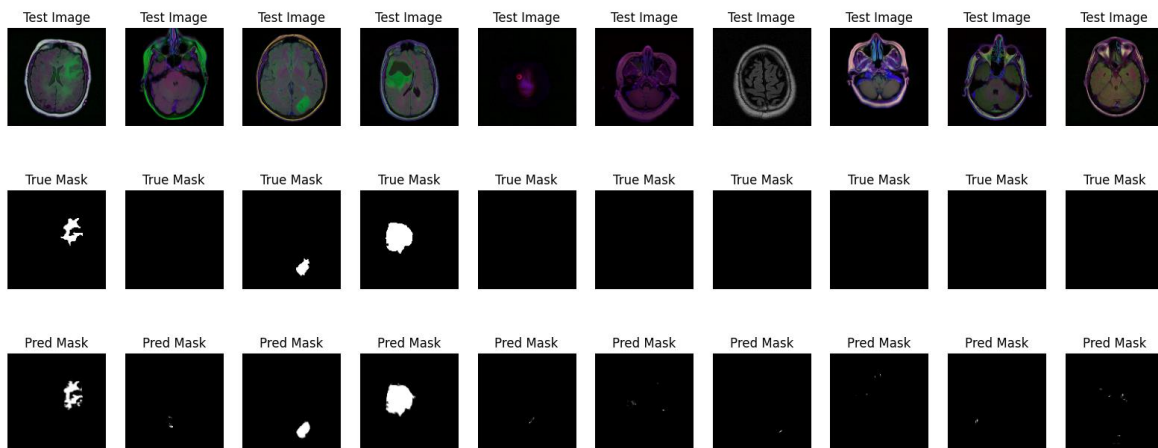
۶-۱. ارزیابی مدل

با ارزیابی عملکرد مدل روی داده‌های تست نتایج زیر بدست آمده است:

Test Loss	Test Accuracy	Test IoU	Test Dice
0.0060	99.8024%	99.5667%	99.7812%

جدول 2. نتایج ارزیابی شبکه‌ی آموزش دیده روی داده‌های ارزیابی

ارقام فوق نمایانگر عملکرد خوب شبکه روی داده‌های ارزیابی می‌باشد. در واقع شبکه‌ی آموزش دیده، روی تصاویر دیده نشده در حین فرایند یادگیری، دقتی حدود 99.8% دارد و با توجه به مقادیر IoU و Dice نیز میتوان نتیجه گرفت که عملکرد شبکه روی این داده‌ها، مطلوب بوده است. همچنین، loss روی داده‌های ارزیابی نیز مقدار کمی است که به معنای خطای عملکردی کمی از سمت مدل در پیش‌بینی ماسک‌ها است. در زیر، نتایج پیش‌بینی مدل روی تعدادی از تصاویر داده‌های ارزیابی را به همراه ماسک اصلی‌شان مشاهده می‌کنیم. همانطور که در این شکل نیز پیداست و همینطور بنا به اعداد گزارش شده در بالا، میتوان نتیجه گرفت شبکه به خوبی آموزش دیده و عملکرد خوبی نیز روی سایر داده‌ها که در فرایند یادگیری مشاهده‌شان نکرده، از خود ارائه می‌دهد.



شکل 9. Maskهای اصلی و پیش‌بینی شده توسط شبکه برای تعدادی از داده‌های ارزیابی به همراه تصاویر اصل

پرسش ۲ – تشخیص موجودات زیر آب

۱-۲. معرفی مدل Faster R-CNN

در اینجا به بررسی مدل Faster R-CNN می‌پردازیم. Faster R-CNN یک مدل پیشرفته در حوزه تشخیص اشیا (Object Detection) است. این مدل از سه قسمت اصلی تشکیل شده است:

1. شبکه عصبی کانولوشنی (CNN) برای استخراج ویژگی‌های تصویر استفاده می‌شود. این بخش شامل لایه‌های کانولوشنی و پولینگ است که ویژگی‌های تصویر ورودی را استخراج و فشرده می‌کند.

2. شبکه پیشنهاد منطقه (RPN) این بخش وظیفه تولید مناطق پیشنهادی (Region Proposals) را دارد. این مناطق مکان‌های احتمالی اشیا در تصویر هستند. RPN یک شبکه کاملاً کانولوشنی است که به طور مستقیم از ویژگی‌های استخراج شده توسط CNN استفاده می‌کند و پیشنهادات مناطق را تولید می‌کند.

3. شبکه تشخیص (Detection Network) این بخش مناطق پیشنهادی را که توسط RPN ارائه شده‌اند، می‌گیرد و به هر کدام یک برچسب دسته‌بندی (مثل انسان، ماشین و غیره) و مختصات دقیق کادر محدود کننده (Bounding Box) اختصاص می‌دهد.

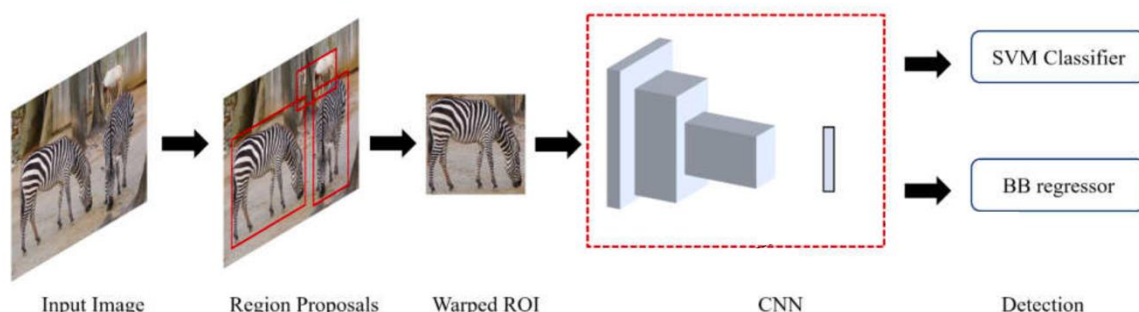
- مزیت اصلی Faster R-CNN نسبت به مدل‌های قبلی این است که فرآیند تولید مناطق پیشنهادی و تشخیص اشیا به طور یکپارچه و همزمان انجام می‌شود که سرعت و دقت مدل را بهبود می‌بخشد.

- به طور خلاصه، Faster R-CNN یک روش کارآمد و دقیق برای تشخیص اشیا در تصاویر است که از شبکه‌های عصبی کانولوشنی و شبکه پیشنهاد منطقه برای بهبود کارایی استفاده می‌کند.

۲-۲. سوالات تشریحی

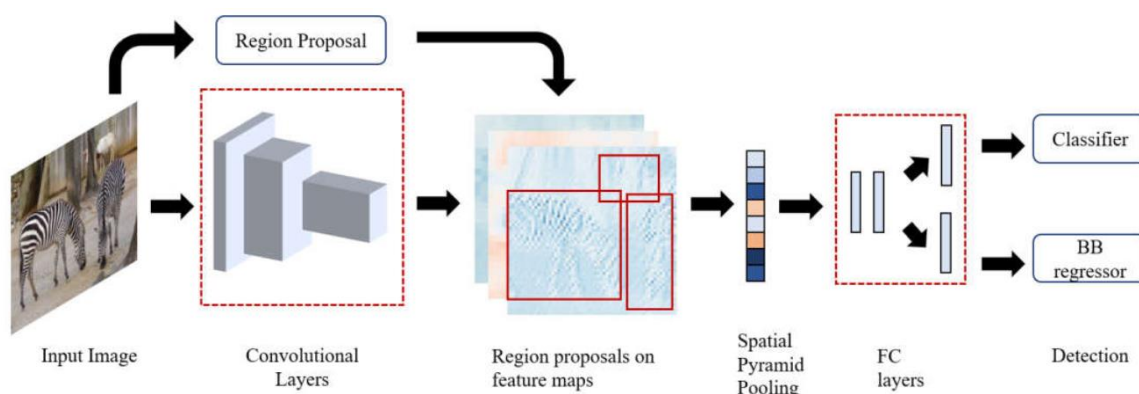
۲-۲-۱. مقایسه‌ی مدل‌های region-based CNNs

- RCNNs: این شبکه اولین بار در سال ۲۰۱۳ معرفی شد. کاربرد اصلی آن برای تشخیص اشیاء مختلف موجود در تصاویر است. این معماری از یک الگوریتم سرچ به نام جستجوی انتخابی استفاده می‌کند که در نتیجه‌ی اجرای این الگوریتم، چیزی حدود ۲۰۰۰ region proposal تولید می‌شود. پس از تغییر اندازه‌ی این proposalها بطوریکه برای ورودی شبکه‌ی کانولوشنی آماده باشند، به CNN ورودی داده می‌شوند و پس از محاسبه‌ی ویژگی‌های CNN، طبقه بندی انجام می‌شود. اما از آنجا که در این شبکه به دنبال تشخیص اشیاء در تصویر هستیم، نیازمند تعیین مکان آنها نیز می‌باشیم. برای این کار، بخش کانولوشنی شبکه برای هر شی، علاوه بر اینکه کلاس آن را مشخص می‌کند، یک bounding box نیز خروجی می‌دهد که مختصات شی مورد نظر در تصویر است. مسئله‌ای که ممکن از آن مواجه شویم، این است که دو یا چند تا از این bounding boxها که شبکه برای اشیاء خروجی می‌دهد، متعلق به یک شی باشند. در این شرایط از non-maximum suppression بهره می‌گیریم. از آنجا که مسائلی که با استفاده از این شبکه‌ها در حال حل آنها هستیم، supervised هستند، این تکنیک، bounding boxها که بیشتری اشتراک و نزدیکی را به ground truth دارد را برای شی در حال بررسی، اعمال می‌کند. محاسبه‌ی میزان این اشتراک نیز به وسیله‌ی معیار IoU صورت می‌گیرد که در سوال قبل مفصلاً به توصیف آن پرداختیم. در زیر، شکل کلی از معماری این شبکه را مشاهده می‌کنید:



شکل ۱۰. معماری شبکه‌ی RCNN

- Fast RCNNs: تفاوت این شبکه با RCNN در این است که قبل از تولید region proposal ها، تصویر تغییر سایز یافته را به شبکه‌ی کانولوشنی می‌دهد و پس از استخراج ویژگی‌ها، جستجوی انتخابی را اعمال می‌کند و region proposal ها را مشخص می‌کند. پس از آن، روی این نواحی spatial pyramid pooling را اعمال می‌کند و خروجی آن را به یک شبکه‌ی fully connected می‌دهد تا کلاس اشیا و bounding box متناظر آن را مشخص کند. امتیاز این شبکه نسبت به RCNN در این است که شبکه‌ی کانولوشنی تنها یک بار مورد استفاده قرار می‌گیرد. اما در RCNN به ازای هر یک از region proposal ها، شبکه‌ی کانولوشنی مورد استفاده قرار می‌گرفت و سرعت آموزش را پایین می‌آورد. اما در این شبکه‌ی جدید، تنها بهش fully connected که پیچیدگی کمتری دارد برای هر یک از region proposal ها تکرار می‌شود. در واقع، در این معماری جدید، مشکل حجم محاسبات و سرعت آموزش تا حدی رفع شده است. در زیر، معماری این شبکه‌ها در حالت کلی را می‌بینیم:

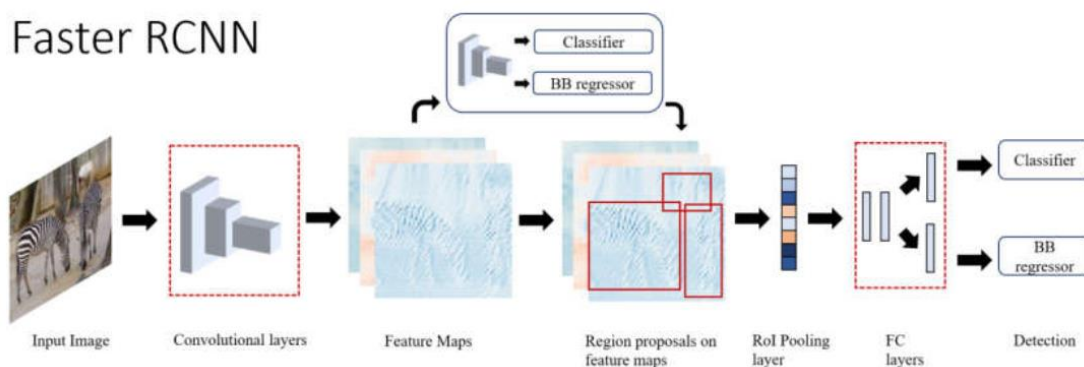


شکل 11. معماری شبکه‌ی Fast RCNN

- Faster RCNN: در دو معماری قبلی از جستجوی انتخابی استفاده می‌شد. اما از آنجا که این الگوریتم جستجو سرعت پایین و محاسبات سنگینی دارد، به دنبال جایگزینی آن هستیم. در Faster RCNN بجای جستجوی انتخابی از Region Proposal Network استفاده می‌کنیم. معماری این شبکه به گونه‌ای است که تصویر ورودی را به بخش کانولوشنی می‌دهد و پس از استخراج feature maps، این نقشه‌ها برای پیدا کردن کلاس و bounding boxهای متناظر با هر شی، به RPN ورودی داده می‌شوند. نحوه‌ی کارکرد RPN نیز به گونه‌ای است که از تعدادی anchor box بهره می‌گیرد که خود این باکس‌ها از bounding boxهای با نسبت‌های متفاوت کمک می‌گیرند تا اشیا را مکان‌یابی کنند. در واقع، روی region proposal های متناظر با

fully feature maps لایه‌ی RoI pooling اعمال می‌شود و خروجی آن نیز به یک بخش connected داده می‌شود تا کلاس و متخصّات هر شی در تصویر مشخص شود.

با توجه به معماری ویژه‌ای که برای RPN در نظر گرفته شده، سرعت این روش نسبت به دو روش قبل بیشتر است. همچنین، دقت را تا حدود ۳ درصد افزایش می‌دهد. در ادامه شکل متناظر با این معماری را مشاهده می‌کنیم:



شکل 12. معماری شبکه‌ی Faster RCNN

۲-۲-۲. مقایسه‌ی مدل‌های one-stage و two-stage

مزایای one-stage: این معماری ها معمولاً ساده‌تر هستند و این سادگی منجر به صرف زمان کمتری برای آموزش شبکه می‌شود. همچنین، از آنجا که bounding box و کلاس اشیا را به صورت مستقیم و در یک گذر پیش‌بینی می‌کنند، سریع‌تر هستند. از این رو این معماری در شرایطی که به real-time performance احتیاج داریم، مثل video analysis و یا autonomous driving، عملکرد بهتری دارند. همچنین، این detectorها در تشخیص اشیا کوچک موجود در تصاویر بهتر عمل می‌کنند.

معایب one-stage: در شرایطی که اشیا موجود در تصاویر کوچک هستند و یا پس‌زمینه پیچیده است، این detectorها معمولاً نسبت به two-stageها دقت پایین‌تری از خود نشان می‌دهند. همچنین، نسبت به two-stageها در مواردی مکان‌یابی اشیا را با دقت کمتری انجام می‌دهند و مکان bounding box ممکن است دچار خطا باشند. یکی دیگر از معایب این تشخیص دهنده‌ها این است که در پیدا کردن روابط بین اجزا و اشیا در تصویر عملکرد ضعیف‌تری دارند و این مسئله می‌تواند کارکرد کلی شبکه را تحت تأثیر قرار دهد.

کاربرد one-stage detectors:

1. رانندگی خودکار (Autonomous Driving) :

در خودروهای خودران، تشخیص اشیا در زمان واقعی ضروری است. آشکارسازهای تکمرحله‌ای به دلیل سرعت بالا می‌توانند در تشخیص سریع و دقیق اشیا مانند عابران پیاده، خودروهای دیگر، علائم راهنمایی و رانندگی و موانع کمک کنند. این اطلاعات برای تصمیم‌گیری‌های لحظه‌ای و جلوگیری از تصادفات بسیار مهم هستند.

2. تشخیص اشیا در زمان واقعی (Real-time Object Detection)

در کاربردهایی که نیاز به پردازش و واکنش سریع دارند، مانند رباتیک و سیستم‌های نظارتی، آشکارسازهای تکمرحله‌ای به دلیل کارایی و سرعت بالا بسیار مفید هستند. این مدل‌ها می‌توانند به سرعت اشیا را در تصاویر ویدئویی شناسایی و دسته‌بندی کنند.

3. واقعیت افزوده (Augmented Reality)

در سیستم‌های واقعیت افزوده، تشخیص سریع و دقیق اشیا در محیط واقعی برای اضافه کردن لایه‌های مجازی بسیار مهم است. آشکارسازهای تکمرحله‌ای می‌توانند به سرعت اشیا را در محیط شناسایی کنند و تجربه واقعیت افزوده را بهبود بخشند.

4. شمارش و ردیابی اشیا (Object Counting and Tracking)

در کاربردهایی مانند شمارش افراد در محیط‌های عمومی، ردیابی خودروها در جاده‌ها یا نظارت بر انبارها، آشکارسازهای تکمرحله‌ای می‌توانند به سرعت و با دقت اشیا را شناسایی، شمارش و ردیابی کنند. این کاربردها در تحلیل‌های آماری و مدیریتی بسیار مهم هستند.

نمونه‌های one-stage detectors:

- YOLO
- SSD
- YOLO2
- Retina Net
- YOLOv3
- Center Net
- EfficientNet
- YOLOv4
- Swin transformer
- YOLOx

مزایای two-stage:

- این تشخیص دهنده‌ها معمولاً دقت بیشتری نسبت به نوع قبلی دارند، به ویژه در مواردی که اشیاء کوچک هستند و یا پس‌زمینه به هم ریخته و نویزی است. به دلیل دو مرحله‌ای بودن، معمولاً نسبت به مواردی نظیر مقیاس، robustness بیشتری دارند. همچنین، bounding box‌های بدست آمده از این معماری‌ها، دقت بهتری دارند و مکان‌یابی اشیاء را به شکل درست‌تری نمایش می‌دهند. همچنین، برخلاف معماری تک مرحله‌ای، روابط بین اشیاء را بهتر پیدا می‌کنند که در مجموع عملکرد تشخیصی را افزایش می‌دهد.

معایب two-stage:

- این معماری نسبت به معماری قبل، پیچیدگی بیشتری دارد و پروسه‌ی آموزش شبکه مدت زمان بیشتری طول می‌کشد. بنابراین، هزینه‌ی انجام محاسبات در این معماری نیز نسبت به معماری قبل، بسیار بیشتر است.

کاربردهای two-stage:

1. تشخیص اشیاء با جزئیات دقیق (Fine-grained Object Detection):
در کاربردهایی که نیاز به شناسایی تفاوت‌های کوچک و جزئیات دقیق بین اشیاء دارند، آشکارسازهای دو مرحله‌ای بسیار موثر هستند. این شامل تشخیص نژادهای مختلف حیوانات، مدل‌های مختلف خودروها، یا انواع مختلف محصولات است. این مدل‌ها به دلیل دقت بالاتر در شناسایی جزئیات، برای این نوع کاربردها مناسب‌تر هستند.
2. شناسایی اشیاء در صحنه‌های پیچیده (Object Recognition in Complex Scenes):
در صحنه‌هایی که شامل تعداد زیادی شیء با تداخلات پیچیده هستند، آشکارسازهای دو مرحله‌ای به دلیل توانایی‌شان در تولید نواحی پیشنهادی و سپس بررسی دقیق هر ناحیه، عملکرد بهتری دارند. این کاربردها شامل تصاویر شهری با تعداد زیادی عابر پیاده، خودروها و سایر اشیاء است.

3. درک صحنه و استدلال مبتنی بر زمینه (Scene Understanding and Contextual Reasoning):

در کاربردهایی که نیاز به درک کلی از صحنه و استدلال بر اساس زمینه دارند، آشکارسازهای دو مرحله‌ای می‌توانند به تفکیک اشیا و تحلیل روابط بین آنها کمک کنند. این شامل تحلیل تصاویر برای استخراج اطلاعاتی مانند تعاملات انسانی، توزیع اشیا در یک صحنه و استدلال درباره موقعیت‌ها و وقایع است.

4. تقسیم‌بندی معنایی (Semantic Segmentation):

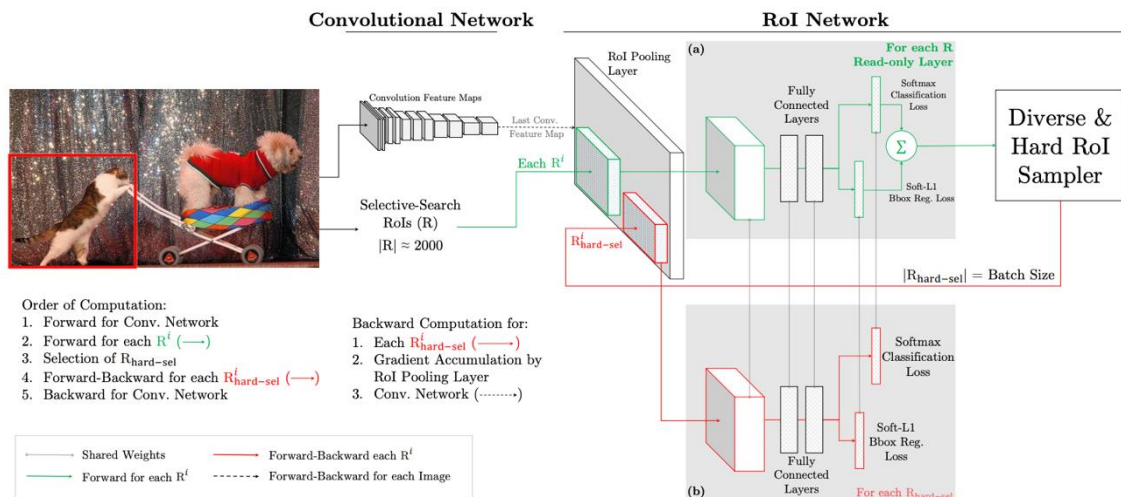
هرچند که تقسیم‌بندی معنایی به طور معمول با روش‌های خاص خودش انجام می‌شود، آشکارسازهای دو مرحله‌ای می‌توانند در تشخیص دقیق اشیا و نواحی خاص در تصاویر کمک کنند. این مدل‌ها می‌توانند نواحی مختلف یک تصویر را با دقت بالا دسته‌بندی کنند که برای کاربردهای تقسیم‌بندی معنایی مفید است.

نمونه‌های two-stage detectors:

- RCNN
- Fast RCNN
- Faster RCNN
- SPP-Net
- FPN
- RFCN
- Mask RCNN

OHEM

این تکنیک برای مسائلی که در آنها به عنوان تشخیص اشیا در تصاویر هستیم، کاربرد دارد. در آموزش یک شبکه‌ی عصبی، مجموعه دادگان معمولاً شامل تعداد زیادی نمونه‌های ساده و تعداد کمی نمونه‌های دشوار است. مدل، از نمونه‌های آسان به سادگی یاد می‌گیرد و با توجه به میزان داده‌های آسان به دشوار، معمولاً تشخیص‌بندی تا حد خوبی نیز صورت می‌گیرد. اما شبکه در مواجهه با نمونه‌های سخت، عملکرد خوبی از خود ارائه نمی‌دهد. OHEM با تمرکز بر تشویق شبکه برای یادگیری نمونه‌های سخت، سعی در بهبود عملکرد آن دارد. این کار را با انتخاب پویای نمونه‌های سخت از بین سایر تصاویر انجام می‌دهد و با اعمال گام‌های backward، منجر به این می‌شود که شبکه روی نمونه‌های سخت نیز متمرکز شود و آنها را نیز یاد بگیرد. کاربرد این تکنیک به گونه‌ای است که پس از اینکه region proposalها برای تصاویر یک دسته آماده شدند، با محاسبه‌ی loss برای آنها نسبت به ground truth موجود در دیتاست، تصاویری که loss بیشتری دارند را در دسته‌های کوچک‌تر به مدل می‌دهد تا در گام‌های بعدی فرایند یادگیری، روی این تصاویر متمرکزتر شود. با این توضیحات، این تکنیک منجر به بهبود کارایی مدل می‌شود و تشخیص اشیا در تصاویر بهتر صورت می‌گیرد. مدل در مواجهه با نمونه‌های سخت، عملکرد بهتری از خود ارائه می‌دهد. در زیر شکل کلی کارکرد این تکنیک را میتوان مشاهده کرد:



شکل 13. نحوه‌ی کارکرد تکنیک OHEM برای مسئله‌ی تشخیص اشیا در تصویر

کاربردهای OHEM:

- Object detection
- Face recognition
- Action recognition
- Medical diagnosis

Soft-NMS:

این تکنیک نیز برای بهبود پیش‌بینی‌های مدل از مکان‌یابی اشیاء در تصاویر استفاده می‌شود و یک نسخه‌ی ارتقا یافته از NMS است که در قسمت قبل اندکی به توصیف آن پرداختیم. همانطور که می‌دانیم، ممکن است مدل برای هر شیء در تصویر تعدادی bounding box تولید کند. درواقع، ممکن است تعدادی از bounding box های تولید شده توسط مدل، متعلق به یک شیء باشند و با هم اشتراک داشته باشند. در NMS سیاست این بود که با استفاده از معیار IoU و تعیین یک threshold، میزان اشتراک هر یک از این باکس‌ها را با ground truth بدست آوریم و سپس باکسی که امتیاز آن بالای آستانه قرار می‌گیرد و بیشترین اشتراک و نزدیکی را به ground truth دارد، به عنوان نتیجه خروجی دهیم و از سایر باکس‌ها چشم‌پوشی می‌کنیم. اما قرار دادن یک آستانه‌ی سخت برای انتخاب بهترین bounding box در مواردی منجر به از دست دادن تشخیص‌های معتبر می‌شود، به ویژه زمانی که امتیازات bounding box ها به هم نزدیک باشند. برای رفع این مشکل در NMS، تکنیک Soft-NMS معرفی شد که به این صورت عمل می‌کند که برای bounding box هایی که امتیاز پایین‌تری دارند، بر اساس میزان هم‌پوشانی‌شان با ground truth، امتیازات آنها را کاهش می‌دهد به طوریکه برای هر باکس، امتیاز باکس‌های هم‌پوشانی را با توجه به یک تابه که به IoU وابسته است، کاهش می‌دهد و هرچه هم‌پوشانی بیشتر باشد، امتیاز را بیشتر کاهش می‌دهد. این تکنیک این تضمین را به ما می‌دهد که باکس‌هایی که امتیاز بالا اما هم‌پوشانی کم دارند، به طول کامل کنار گذاشته نمی‌شوند و صرفاً تاثیر آنها کاهش می‌یابد. لازم به ذکر است که این تکنیک برخلاف ورژن قدیمی خودش، فاقد یک آستانه‌ی سخت است و با اعمال آن، نتایج دقت بیشتری دارند و منجر به بهبود عملکرد مدل در تشخیص اشیاء موجود در تصاویر می‌شود.

کاربردهای soft-NMS:

- Object detection
- Face detection
- Surveillance systems
- Robotics

:GIOU

این تکنیک یک ورژن جدید از معیار IoU است که قبلاً به توصیف آن پرداخته بودیم و برای سنجش عملکرد مدل استفاده می‌شد. این معیار در مواردی مشکل‌ساز است، خصوصاً وقتی که بین bounding boxها هم‌پوشانی وجود نداشته باشد. برای رفع این مشکل، GIOU معرفی شد که به این صورت عمل می‌کند: ابتدا مقدار IOU را محاسبه می‌کند. سپس کوچک‌ترین باکسی که شامل باکس پیش‌بینی شده و ground truth می‌شود را پیدا می‌کند. سپس مساحت این باکس جدید و همین‌طور مساحت حاصل از اجتماع دو باکس پیش‌بینی شده و ground truth را محاسبه می‌کند. سپس با استفاده از فرمول زیر، مقدار GIOU را محاسبه می‌کند:

$$GIoU = IoU - \frac{C - Union}{C}$$

معادله 4. فرمول محاسبه‌ی GIOU

در فرمول فوق، C همان مساحت کوچک‌ترین باکس شامل ground truth و باکس پیش‌بینی شده توسط مدل و Union مساحت اجتماع این دو باکس است. با استفاده از این ورژن جدید، این اطمینان به ما داده می‌شود که نه تنها هم‌پوشانی باکس‌ها با هم را در نظر گرفته ایم، بلکه میزان فاصله‌ی دو باکس از هم نیز در نظر گرفته شده است. این کار منجر به بهبود عملکرد مدل در حین یادگیری می‌شود. در واقع این تکنیک گرادیان‌های حامل اطلاعات بیشتری را برای ما فراهم می‌کند که به همگرایی بختری منجر می‌شود. همچنین، منجر به دستیابی به دقت‌های بالاتر در تشخیص اشیا در تصاویر می‌شود، خصوصاً زمانی که IoU صفر باشد.

:کاربردهای GIOU

- Object detection
- Localization
- Autonomous vehicles
- Medical imaging

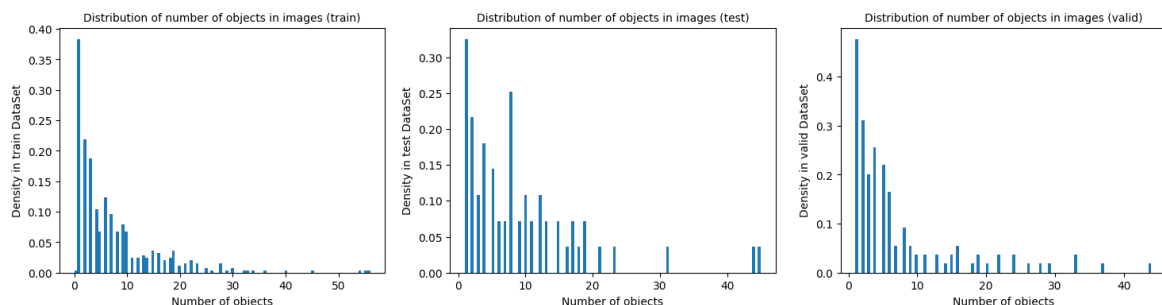
۳-۲. معرفی مجموعه دادگان

مجموعه داده مورد استفاده در این مقاله، Underwater Object Detection نام دارد. این مجموعه دادگان شامل ۶۳۸ تصویر از موجودات زیر آب است که در ۷ کلاس طبقه بندی می‌شوند. (۴۴۸ داده آموزش، ۱۲۷ داده ارزیابی و ۶۳ داده تست). همچنین همراه با این تصاویر، جعبه‌های مرزی موجودات حاضر در هر تصویر موجود است.

۳-۲-۱. EDA و پیش پردازش دادگان

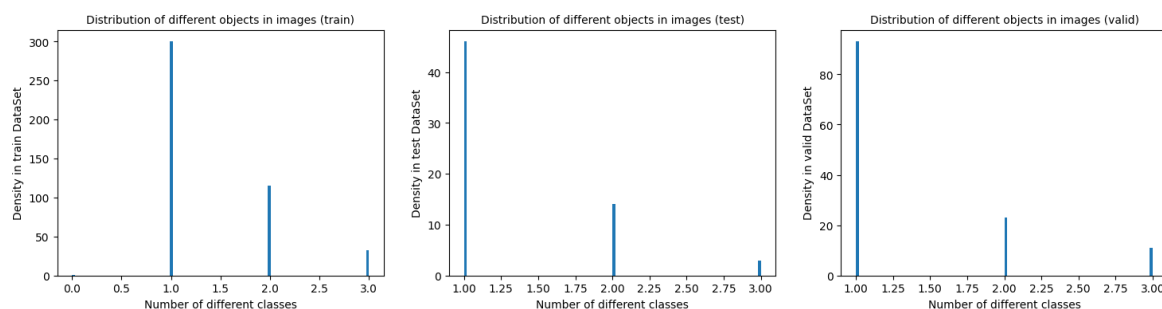
در این بخش قصد داریم به بررسی داده‌های در اختیار بپردازیم. در ابتدا لازم است که بیان کنیم، اندازه تصاویر در این مجموعه داده، متفاوت از هم دیگه هستند و همه آن‌ها در یک اندازه نیستند. برای مثال تصاویر با اندازه‌های ۷۶۸ در ۱۰۲۴، ۱۰۲۴ در ۷۶۸، ۷۶۸ در ۷۶۸ و ... است. بنابراین نیاز است که همه تصاویر را به اندازه یکسان ببریم. این کار را در قسمت DataLoader و با استفاده از transforms از کلاس Albumentations جلوتر انجام خواهیم داد. همچنین می‌توانیم برای سبک‌تر شدن محاسبات، تصاویر را نرمالایز کنیم (نرمالایز خاص مدل backbone مورد استفاده). و پس از آن، داده تصاویر را به جای آراییه، در تانسور ذخیره کنیم.

حال به بررسی کلاس‌ها و جعبه‌های مرزی داده شده و فراوانی و ... آن‌ها می‌پردازیم. این ویژگی‌های مجموعه دادگان در اختیار را در نمودارهایی خواهیم دید.



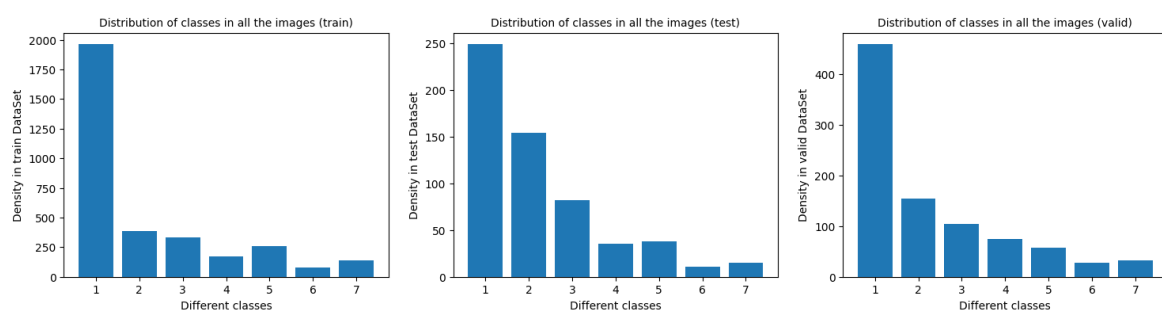
شکل 14. نمودار مربوط به فراوانی موجودات در هر تصویر

در شکل ۱۴، نمودارهای مربوط به فراوانی موجودات در هر تصویر را می‌بینیم. عموماً هر تصویر بین ۰ تا ۲۰ موجود را شامل می‌شود. این موضوع برای هر سه مجموعه آموزش، اعتبارسنجی و تست یکسان است. بنابراین در این قسمت نیازی به اقدام خاصی نبوده و داده‌ها به صورت یکسانی توزیع شده‌اند.



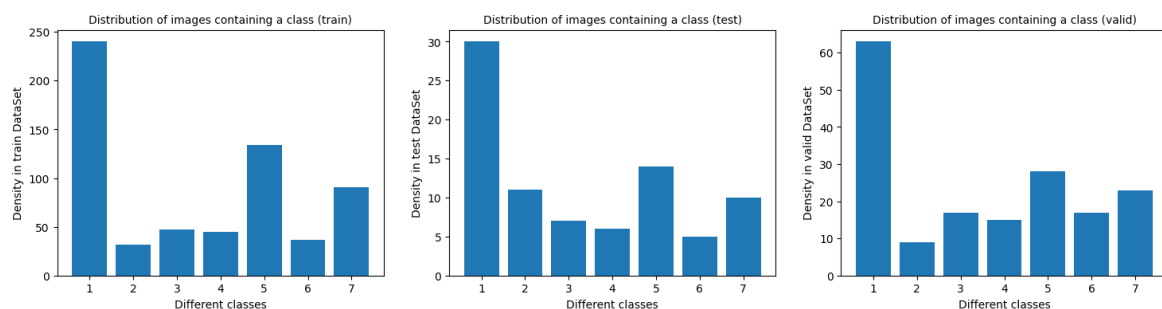
شکل 15. نمودار مربوط به فراوانی کلاس‌ها در هر تصویر

در شکل ۱۵، نمودار مربوط به فراوانی نوع موجودات داخل هر تصویر موجود است. همانطور که مشاهده می‌کنیم، در هر تصویر بین ۱ تا ۳ کلاس نهایتاً موجود است که این مورد برای هر سه مجموعه آموزش، اعتبارسنجی و تست یکسان است. در این صورت نیاز به اقدامی در این مورد نیست. البته قابل ذکر است که در مجموعه آموزش، یک تصویر موجود است که هیچ جعبه مرزی‌ای ندارد. در واقع هیچ موجودی در آن تشخیص داده نشده.



شکل 16. توزیع انواع موجودات در کل مجموعه‌ها

همانطور که در شکل ۱۶ مشاهده می‌کنیم، این نمودارها تعداد انواع موجودات در هر یک از مجموعه داده‌گان آموزش، اعتبارسنجی و تست را نشان می‌دهد. با توجه به نمودارها، به طور مشخص در همه مجموعه‌ها از موجود اول که در واقع بیانگر 'fish' است، از دیگر موجودات بیشتر موجود است. بعد از آن به طور تقریبی تعداد موجودات به ترتیب در همه مجموعه‌ها تعداد نزولی خواهد داشت. در این بخش می‌توان تصاویری که شامل موجودات با فراوانی کمتر هستند را بیش‌تر کرد تا مدل توانی تشخیص همه موجودات را به صورت یکسان بدست آورد و در واقع همه آن‌ها را به خوبی یاد بگیرد. البته این کار برای کاربرد مدل در دنیای واقعی تاثیر به سزایی دارد، اما در اینجا، چون توزیع موجودات تقریباً یکسان است، نیازی به انجام این کار نخواهد بود و مدل دقت خوبی ارائه خواهد داد (که خوب به طور مشخص این دقت بالا در کاربردهای واقعی، به این صورت نبوده و بسیار پایین‌تر خواهد بود).



شکل ۱۷. نمودار مربوط به توزیع تعداد تصاویر شامل هر کلاس

همانطور که در شکل ۱۷ مشاهده می‌کنیم، این نمودارها بیانگر تعداد عکس شامل هر کدام از ۷ موجود ما است. به طور مشخص دوباره تعداد تصاویر شامل کلاس اول، ینی ماهی، از مابقی کلاس‌ها بیشتر است. بعد از آن، تعداد تصاویر از کلاس‌های ۵ و ۷، یعنی کوسه و سفره‌ماهی به ترتیب بیشتر است. بعد از آن‌ها مابقی کلاس‌ها در حدود یکسانی تصویر دارند. دوباره در اینجا مانند حالت قبل، بهتر است که جوری داده‌ها را افزایش دهیم که تعداد تصاویر از هر کلاس تقریباً یکسان باشد تا در شرایط و کاربردهای واقعی، مدل ما بتواند به خوبی هر موجود از کلاس‌های مختلف را تشخیص دهد. اما در اینجا و صرفاً برای آموزش مدل و بالا بردن دقت، نیازی به انجام این کار نخواهد بود.

۲-۳-۲. تقویت داده

در این بخش به تقویت داده و یا Augmentation می‌پردازیم. برای تقویت داده، ما از کتابخانه Albumentations استفاده کردیم. این کتابخانه نسبت به transforms از torch، یک سری خوبی‌های بیشتری دارد که در تقویت داده در این مسئله به ما کمک فراوان می‌کند.

قبل از بیان برتری Albumentations نسبت به transforms، لازم است ایرادی که در transforms موجود است و بسیار مدت زمان زیادی را برای حل این مشکل با استفاده از آن صرف کردم (در آخر بی‌ثمر بود و از Albumentations استفاده کردم)، را ببینیم.

یکی از موارد بسیار مهمی که برای تقویت داده در مسئله تشخیص (detection) وجود دارد، این است که در صورت استفاده از عملیاتی مانند معکوس کردن یا چرخاندن و ... که محل جعبه‌های مرزی (bounding box) تغییر می‌کند، هندل کردن محل جدید آن‌ها است.

به نظر مشخص است که transform قادر به انجام این کار نبود و تنها تصویر را جابه‌جا می‌کرد و با جعبه‌های مرزی کاری نداشت و آن‌ها را به روز نمی‌کرد. با جست‌جوی زیاد برای انجام آگمنتیشن و کلی تلاش برای استفاده از همان transforms، در آخر کتابخانه Albumentations را پیدا کردم که جعبه‌های ما را نیز به روز می‌کرد.

با استفاده از این کتابخانه، تقویت داده را به این صورت انجام دادم. ابتدا روش‌های تقویت داده اختصاصی برای داده‌های آموزش را بررسی کنیم.

- GaussNoise: در این قسمت، یک نویز گاوسی با میانگین ۰ و واریانس بین ۱۰ تا ۵۰، به تصویر اصلی اضافه می‌شود
- ColorJitter: تغییرات تصادفی در روشنایی، کنتراست، اشباع و هيو (رنگ) تصویر ایجاد می‌کند. پارامترهای brightness, contrast, saturation و hue محدوده تغییرات را مشخص می‌کنند. $p=0.5$ احتمال اعمال این تبدیل را نشان می‌دهد.
- RGBShift: تغییرات تصادفی در مقادیر کانال‌های رنگی قرمز، سبز و آبی ایجاد می‌کند. r_shift_limit ، g_shift_limit و b_shift_limit محدوده تغییرات را مشخص می‌کنند. $p=0.5$ احتمال اعمال این تبدیل را نشان می‌دهد.

همه این تبدیل‌ها، با احتمال ۵۰ درصد به صورت جداگانه اتفاق می‌افتند.

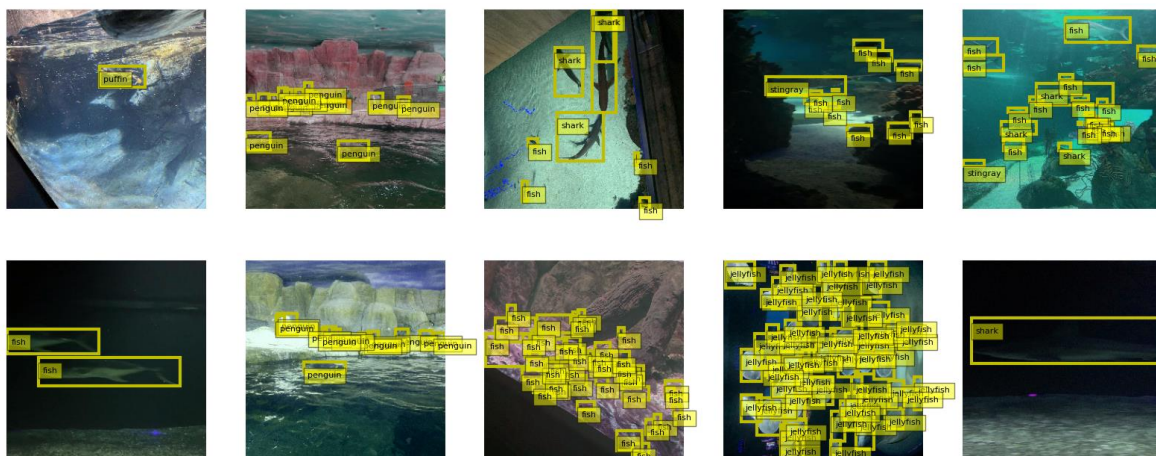
حال تغییراتی که برای داده‌های آموزش، اعتبارسنجی و تست انجام می‌شود را بررسی می‌کنیم.

- **Resize:** در این قسمت، سایز تصاویر به ۲۵۶ در ۲۵۶ تبدیل می‌شود تا همه آن‌ها اندازه یکسانی داشته باشند.

- **ToTensorV2:** در نهایت، تصویر را به فرمت PyTorch Tensor در می‌آورد.

همچنین می‌خواستم روش‌های مانند Flip و Rotate را انجام دهم، اما این روش‌ها حتی با کتابخانه Albumentations نیز به خوبی عمل نکرده و در برخی موارد جعبه‌های مرزی از بین میرفتند. بنابراین به همین روش‌های گفته شده بسنده کردم. چرا که خود تصاویر به گونه‌ای هستند که تا حد خوبی چرخش و معکوس کردن را خود به خود انجام دهند. بنابراین تغییراتی در خود تصاویر به خودی خود کافی خواهد بود و نیازی به دیگر روش‌های مانند Flip و ... نخواهد بود.

نتیجه این تقویت داده را برای ۱۰ تصویر (قسمتی از یک batch) می‌توانیم در شکل زیر مشاهده کنیم.



شکل 18. نمونه‌هایی از تقویت داده

۲-۳-۳. ساخت دیتالودر

دیتالودر ما از دو بخش تشکیل شده. که به هر کدام از آن‌ها می‌پردازیم:

- کلاس دیتاست: در واقع این کلاس وظیفه آماده سازی هر کدام از داده‌ها در یک batch را به عنوان ورودی به مدل به عهده خواهد داشت. همچنین همانطور که قبلاً گفتیم، تقویت داده در این قسمت و در واقع قبل از رسیدن به مدل و در آن واحد انجام می‌شود.
 - متد `__init__`: در این قسمت آدرس تصاویر و فایل جعبه‌های مرزی مربوط به هر کدام از تصاویر را ذخیره می‌کنیم. علاوه بر آن‌ها عملیات تعریف شده برای تقویت داده را در این قسمت تنظیم می‌کنیم.
 - متد `__len__`: این تابع تنها تعداد تصاویر موجود در آدرس داده شده را برمی‌گرداند.
 - متد `__getitem__`: این تابع در واقع وظیفه انتخاب تصاویر و انجام عملیاتی بر روی آن‌ها تا قبل از قرار گیری در یک batch و ارسال به مدل را دارد.
- در ابتدا تصویر از آدرس موجود گرفته شده و سپس جعبه‌های مرزی و برچسب آن‌ها از آدرس موجود را دریافت می‌کنیم. می‌دانیم که در این دیتاست، جعبه‌های مرزی به این صورت تعریف می‌شوند: (به فرمت COCO)

$$bounding\ box = [x_{center}, y_{center}, width, height]$$

اما برای استفاده آن‌ها را به صورت زیر در می‌آوریم: (به فرمت Pascal_voc)

$$bounding\ box = [x_{min}, y_{min}, x_{max}, y_{max}]$$

سپس تقویت داده (آگمنتیشن) تعریف شده را بر روی تصویر، جعبه‌های مرزی و برچسب مربوط به هر جعبه مرزی انجام می‌دهیم.

بعد از آن به عنوان مقدار برگشتی، تصویر آگمنت شده و یک دیکشنری از جعبه‌های مرزی و برچسب آن‌ها را برمی‌گردانیم.

- تابع `collate_fn`: این تابع در حقیقت وظیفه جمع‌آوری یک `batch` با استفاده از کلاس دیتاست دارد. به این صورت که به تعداد `batch` تعیین شده، تصاویر، جعبه‌های مرزی و برچسب‌ها را به یک‌دیگر اضافه می‌کند و به این صورت برمی‌گرداند:

```
return torch.stack(images), targets
```

که `targets` در واقع یک لیست از دیکشنری‌ها که در هر کدام جعبه‌های مرزی و برچسب‌های مربوط به هر تصویر است را شامل می‌شود.

همچنین یکی دیگر از وظایف مهم دیگر این تابع، این است که به گونه‌ای تعداد جعبه‌های درون یک `batch` را تغییر دهد که همه آن‌ها به یک تعداد جعبه داشته باشند. طبیعتاً این اتفاق زمانی به صورت منطقی اتفاق می‌افتد که تعداد همه آن‌ها به اندازه بیشترین تعداد جعبه برای یک تصویر در آن `batch` خواهد بود.

در این صورت با استفاده از این توابع و تعریف دیتالور به صورت زیر، می‌توانیم از داده‌ها به صورت آنی و بدون ذخیره کردن آن‌ها در مموری استفاده کنیم. همچنین در این بین آگمنتیشن نیز اتفاق خواهد افتاد.

```
train_set = Aquarium(root_dir=root_dir, transform=transform)
train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True,
collate_fn=collate_fn)
```

۲-۴. تعریف مسئله

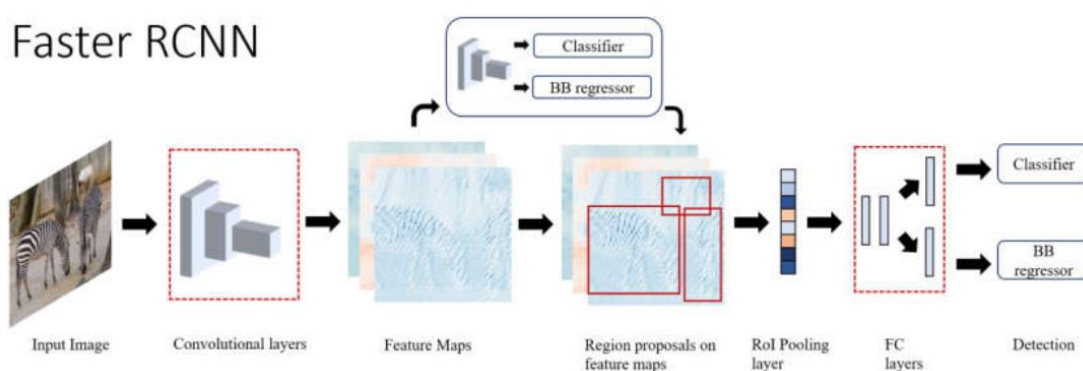
در این مسئله قرار است با استفاده از مدل Faster R-CNN که یک مدل region-based CNNs و همچنین two stage است، مدلی طراحی کنیم که بتواند با استفاده از دیتاست داده شده، ۷ موجود زیر آب که عبارتند از:

- Fish
- Jellyfish
- Puffin
- Shark
- Starfish
- Stingray

را از یکدیگر و همچنین از محیط تشخیص دهیم. در واقع detect کنیم.

۲-۴-۱. طراحی معماری Faster R-CNN

همانگونه که با معماری Faster R-CNN آشنا هستیم و نمونه‌ای از آن را می‌توانیم در شکل زیر ببینیم، قسمت‌های مختلف آن را که پیاده‌سازی شده به ترتیب توضیح خواهیم داد.



شکل 19. نمای کلی Faster RCNN

- بخش Feature Extractor:

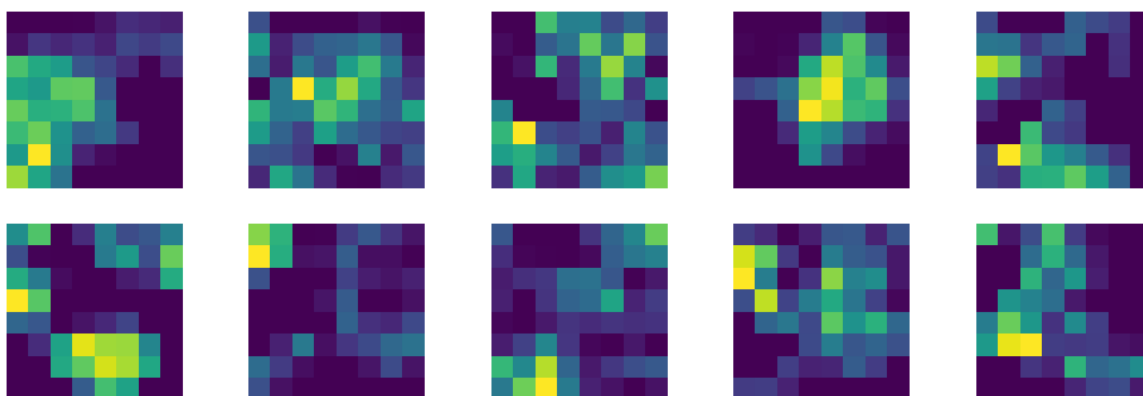
در این قسمت در حقیقت قسمتی از مدل طراحی می‌شود که با استفاده از آن، بتوانیم یک سری ویژگی که در شبکه‌های کانولوشنی در واقع همان Feature Map‌های ما هستند را بدست آوریم. در این قسمت معمولاً از مدل‌های معروف که روی دیتاست‌های بزرگ‌تر و جامع‌تر آموزش دیده‌اند به عنوان مدل backbone استفاده می‌کنیم تا این Feature Map‌ها را بدست آوریم.

در اینجا طبق خواسته مسئله از مدل ResNet101 که از قبل آموزش دیده، به عنوان backbone و برای Feature Extractor استفاده می‌کنیم.

کلاس FeatureExtractor تعریف شده شامل دو تابع است که به توضیح آن‌ها می‌پردازیم.

- `__init__`: در این قسمت resnet101 آموزش دیده را از مدل‌های کلاس torchvision فراخوانی کرده و به عنوان backbone ست می‌کنیم.
- `forward`: در این تابع یک batch ورودی گرفته و آن را به مدل resnet101 می‌دهیم تا Feature Map‌های بدست آمده را برگرداند.

همچنین Feature Map‌های بدست آمده برای یکی از تصاویر دیتاست با استفاده از ResNet101 به صورت زیر بدست آمده. همچنین قابل ذکر است که همه آن‌ها سایز ۳۲ در ۳۲ دارند.



شکل 20. Feature Map‌های بدست آمده از ResNet101

• بخش ProposalModule:

این بخش در اصل وظیفه ایجاد region proposal را با استفاده از classification scores و regression offsets برای تعداد از جعبه‌های مرزی پیشنهادی دارد. در مرحله آموزش، این قسمت بر روی جعبه‌های مرزی مثبت (positive anchor boxes) تمرکز می‌کند. در صورتی که در زمان اعتبارسنجی، داده‌های مورد نیاز برای بررسی بیشتر فراهم می‌کند.

در این کلاس چند تابع وجود دارد که به تفسیر آن‌ها می‌پردازیم.

○ `__init__`: در این قسمت دو head یکی برای تسک classification و دیگری برای تسک regression که برای بدست آوردن برچسبها و جعبه‌های مرزی به ترتیب استفاده می‌شوند، را تعریف می‌کنیم. هر کدام آنها شامل:

- یک لایه کانولوشنی با تابع فعال‌ساز ReLU
- یک لایه Dropout
- یک لایه دیگر کانولوشنی

است.

○ `forward`: در این قسمت ورودی‌ها به هر کدام از بخش‌های طبقه‌بندی و رگرشن داده شده و سپس خروجی تولید شده از آنها بازگردانده می‌شوند. اگر در مرحله آموزش باشیم، با استفاده از positive anchor box های بدست آمده، proposal های خود را بدست می‌آوریم و همه آنها را برمی‌گردانیم.

- بخش `ClassificationModule`: این قسمت در واقع بعد از قسمتی است که proposal های ما روی Feature Map ها تصویر شده‌اند. کاری که این بخش انجام می‌دهد این است که، این proposal ها را گرفته و آنها را برای classification و رگرشن مربوط به جعبه‌های مرزی آماده می‌کند.

قسمت‌های مختلف آن را توضیح می‌دهیم.

○ `__init__`: در این بخش ابتدا RoI Align آن را تعیین می‌کنم. انجام این کار نیاز است چرا که proposal های اندازه‌های متفاوتی دارد و برای وارد کردن آنها به head های مربوط به classification و regression، نیاز داریم که همه آنها در یک اندازه باشند.

سپس یک Average pooling خواهیم داشت تا خروجی‌های RoI را کاهش سایز دهیم در حالی که عمق Feature Map ها را بدست آوریم. بنابراین حجم داده‌ای که منتقل می‌شود کاهش خواهد یافت و از طرفی تمرکز بر روی ویژگی‌های مهم‌تر خواهد بود.

بعد از آن یک سری لایه Fully connected تعریف می‌کنیم. در این قسمت به تعداد خروجی‌های RoI pooling (به صورت flat شده در واقع) در لایه اول ورودی گرفته، سپس یک لایه Dropout خواهیم داشت و بعد از آن دوباره یک لایه Fully connected که به تعداد کلاس‌های ما خروجی دارد. (برای کارکرد بهتر مدل، کلاس‌ها را یکی بیشتر تعریف می‌کنیم به این صورت که کلاس دیگر، در واقع background در نظر گرفته می‌شود).

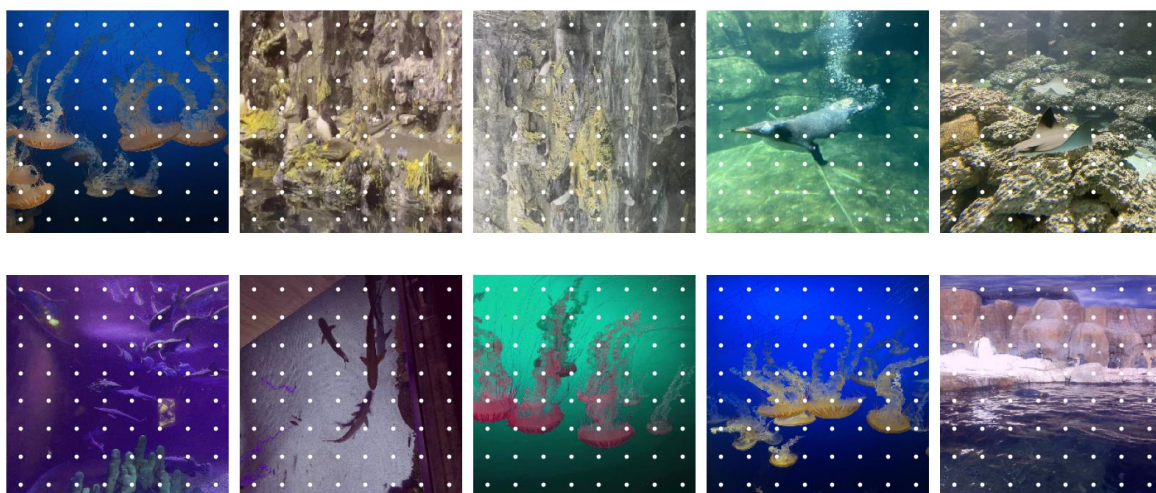
- forward: در این قسمت در واقع ورودی را به هر کدام از این بخش‌ها که در قسمت قبل تعریف کردیم، به ترتیب پاس داده و خروجی آن را به لایه بعدی می‌دهیم.
- اگر در فاز آموزش باشیم، خطا (loss) مربوطه را نیز محاسبه کرده (از خطا cross entropy استفاده می‌کنیم) و بر می‌گردانیم.
- در واقع این بخش نهایی برای classification نهایی مدل ما خواهد بود.
- بخش RegressionModule: در واقع این بخش برای بدست آوردن جعبه‌های مرزی خواهد بود. به این صورت که proposalها را ورودی گرفته و در نهایت جعبه‌های مرزی را خروجی می‌دهد. حال متدهای تعریف شده برای آن را توضیح می‌دهیم.
- __init__: در اینجا مانند قسمت ClassificationModule، یک RoIAlign و یک Average pooling پشت هم تعریف می‌کنیم. سپس یک مدل fully connected بعد از آن با دو لایه تعریف می‌کنیم که در لایه اول به تعداد ویژگی‌های RoI ورودی می‌گیرد. بعد از آن یک لایه Dropout قرار می‌دهیم و سپس یک لایه دیگر fully connected که تعداد خروجی آن ۴ است قرار می‌دهیم. دلیل انتخاب عدد ۴، تعداد مقادیر لازم برای نشان دادن جعبه‌های مرزی است.
- forward: در این قسمت نیز مانند متد forward برای ClassificationModule عمل می‌کنیم و ورودی را به ترتیب به لایه‌های تعریف شده می‌دهیم.
- خود مدل TwoStageDetector: در واقع این قرار است خود مدل دو مرحله‌ای ما باشد که برای تسک detection استفاده خواهیم کرد. حالا به تفسیر قسمت‌های مختلف این کلاس می‌پردازیم:
- __init__: در این قسمت بخش RPN (Region Proposal Network) مدل که بعداً و در قسمت خودش راجع به آن صحبت خواهیم کرد را ست می‌کنیم. سپس بخش classifier مدل را همان ClassificationModule قرار خواهیم داد.
- forward: در این بخش با استفاده از RPN و بخش طبقه بندی، loss مدل را با ورود یک batch محاسبه کرده و آن را برمی‌گردانیم. این قسمت برای بخش آموزش مدل استفاده می‌شود.
- Inference: این بخش برای بخش استفاده از مدل است. در ابتدا با الگوریتم NMS تعدادی از proposalها را انتخاب می‌کند. لازم به ذکر است که این proposalها از RPN

بدست می‌آیند. سپس با استفاده از classifier و میزان اطمینان آن از object بودن در ورودی، آن را با در نظر گرفتن یک احتمال (استفاده از تابع فعال‌ساز softmax) خروجی می‌دهیم.

بنابراین تا به حال قسمت‌های مختلف از یک Faster R-CNN به عنوان یک مدل دو مرحله‌ای (Two Stage) را پیاده‌سازی کرده‌ایم. اگر دوباره به شکل ۱۹ دقت کنیم، می‌بینیم که تنها بخش RPN یا در واقع Region Proposal Network باقی‌مانده این بخش وظیفه map کردن proposal‌های یا درواقع همان جعبه‌های مرزی اولیه روی Feature Map را دارد (نه روی تصویر اصلی، این بخش در قسمت دیگری انجام می‌شود). که بعد از آن نیز از RoI pooling و ... استفاده کرده و سپس با استفاده از دو head برای رگرشن و طبقه‌بندی، تسک مورد نظر برای detection را انجام می‌دهیم.

۲-۴-۲. طراحی Region Proposal Network

در ابتدا به بررسی anchor point‌ها بپردازیم. Anchor point‌ها را به گونه‌ای قرار می‌دهیم که به فاصله یکسان و مساوی روی تصاویر پخش بشوند. چون تصاویر ما همه ۲۵۶ در ۲۵۶ هستند، به نظرم خوب بود که در کل ۶۴ نقطه انکر داشته باشیم. یعنی در هر ردیف و ستون، ۸ نقطه با فواصل مساوی. نمونه‌ای از توزیع این نقاط را روی تصاویر دیتاست در زیر می‌توانیم مشاهده کنیم.



شکل ۲۱. توزیع anchor points

حال باید به دنبال یک روش برای پیدا کردن anchor box های با سایز مناسب برای دیتاست خود باشیم، چرا که با داشتن anchor box با سایز مناسب، می‌توانیم خیلی سریع‌تر و با دقت بهتری اشیا داخل تصاویر را پیدا کنیم. بنابراین تعداد و سایز آن‌ها از می‌تواند از اهمیت نسبتاً بالایی در بحث عملکرد مدل، برخوردار باشد.

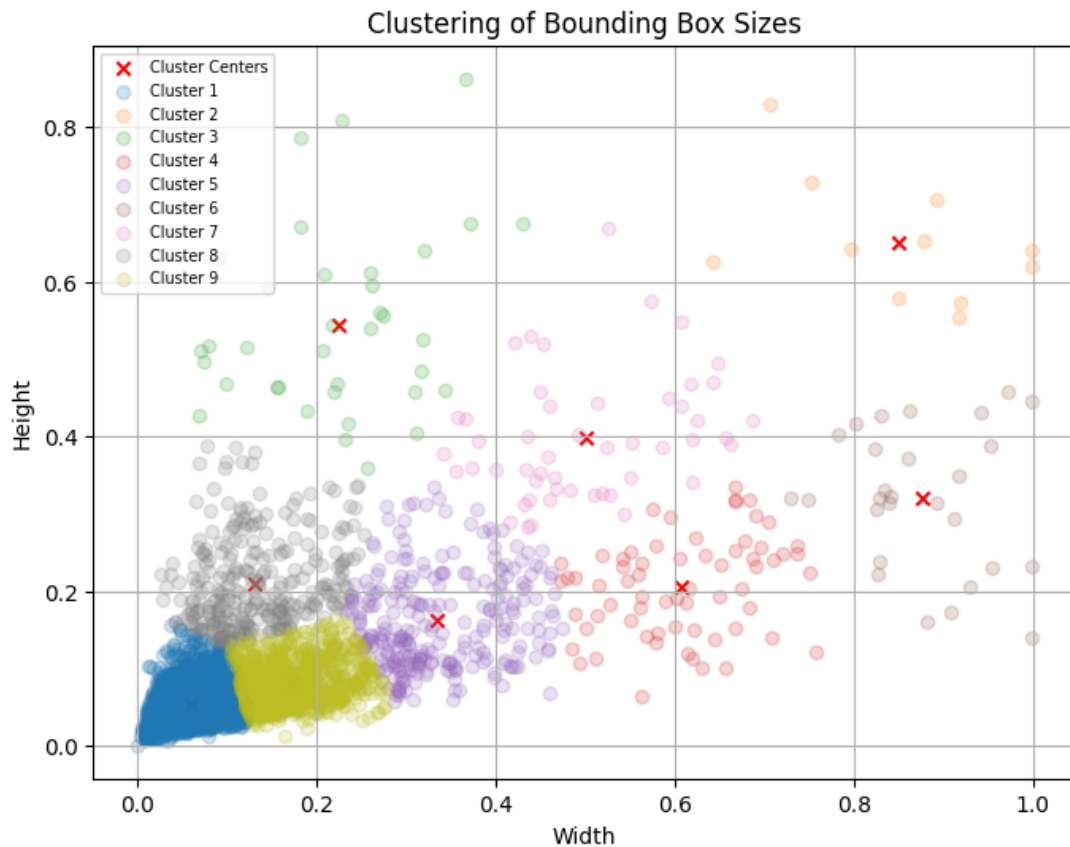
روشی که روش مقاله وجود دارد، k-means++ است، به این صورت که در ابتدا یک سری نقاط مرکزی برای سایز جعبه‌های خود به صورت رندوم انتخاب می‌کنیم که تعداد آن‌ها به اندازه تعداد کل نقاط (در واقع جعبه‌های موجود) است. سپس مراکز را به ترتیب و به تعداد کلاسترهای مورد نظر، انتخاب می‌کنیم به صورتی که بیشترین کمترین فاصله از سایر نقاط را داشته باشند.

بعد از آن، این بار الگوریتم k-means را با نقاط اولیه‌ای که بدست آوردیم اجرا می‌کنیم تا نقاط مرکزی به بهینه‌ترین حالت ممکن بدست آیند. نتیجه‌ای که برای سایزهای anchor box های نرمالایز شده خود داریم به این صورت است:

Anchor Boxes:

```
[[0.06016524 0.05441015]
 [0.8503196 0.64962121]
 [0.22419343 0.5443703 ]
 [0.60765862 0.20632102]
 [0.33450413 0.16245933]
 [0.8757983 0.31955683]
 [0.50130859 0.39836372]
 [0.13179867 0.21062988]
 [0.173367 0.08254046]]
```

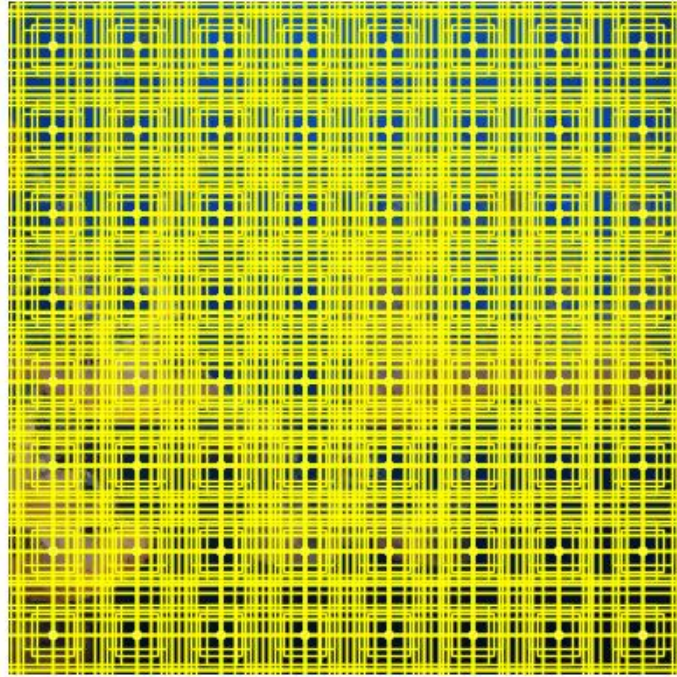
همچنین توزیع سایزهای جعبه‌های مرزی و همچنین نقاط مرکزی تعیین شده با استفاده از الگوریتم k-means++ به این صورت است:



شکل 22. توزیع سایز جعبه‌های مرزی و نقاط مرکزی به دست آمده

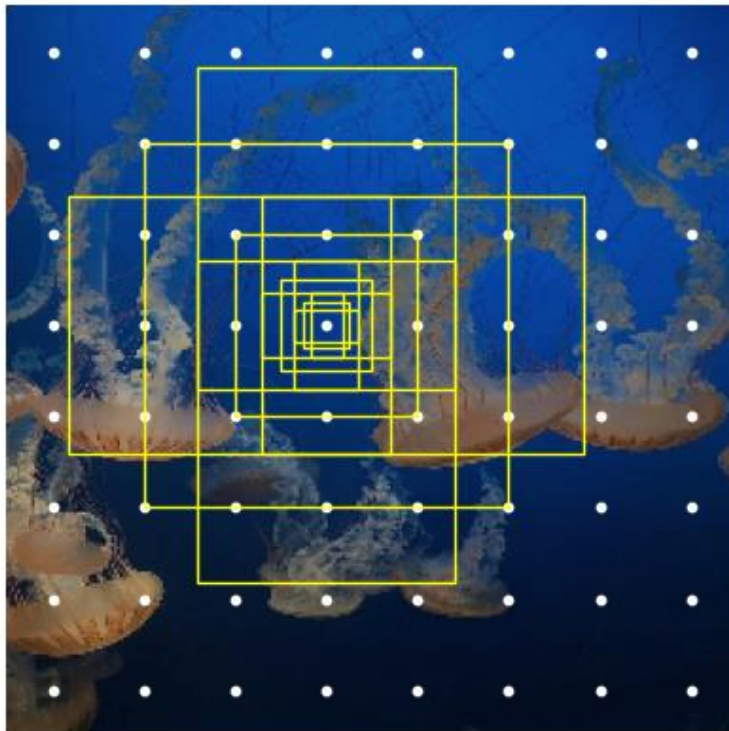
البته به دلیل اینکه ممکن است سایز موجودات موجود در دیتاست آموزش، خیلی متفاوت‌تر با سایز موجوداتی که در مجموعه‌های اعتبارسنجی و تست هستند، باشد، من از این روش استفاده نکردم و روش خیلی ساده‌تری برای انتخاب سایزها در نظر گرفتم تا اگر مجموعه آموزش از دیگر مجموعه‌ها خیلی متفاوت بود، با مشکل مواجه نشویم. اصولاً اتکا به ویژگی‌هایی که خیلی عمومی نبوده و ممکن است تنها در مجموعه آموزش دیده شود، کار خیلی درستی نیست. البته در اینجا شدت خیلی پایین‌تر بود و صرفاً برای اطمینان این روند در پیش گرفته شده.

بنابراین anchor box ها به این صورت با سایزهای ۱۶، ۳۲، ۶۴، ۱۲۸ و با ضرایب ۰.۵ و ۱ و ۲ ساخته می‌شوند. بنابراین در کل ۱۲ anchor box خواهیم داشت. در شکل زیر می‌توانیم همه anchor box های پیشنهاد شده را روی یکی از تصاویر این مجموعه ببینیم.



شکل 23. **Anchor box** های روی یکی از تصاویر

همچنین می‌توانیم **anchor box** های پیشنهادی را تنها برای یکی از **anchor point** ها روی همان تصویر در شکل ۲۴ ببینیم.



شکل 24. **Anchor boxes** برای یک **anchor point**

حال به بررسی خود RPN می‌پردازیم. همان‌طور که می‌دانیم این بخش برای map کردن proposalها بر روی Feature map ها استفاده می‌شود. در واقع وظیفه تولید Region Proposal ها را به عهده دارد.

- `__init__`: در این بخش ابتدا Feature Extractor را یک نمونه‌ای از همان کلاس مربوط به این کار که از ResNet101 استفاده می‌کند، قرار می‌دهیم. سپس Proposal Module آن را همان کلاسی که در بخش قبل تعریف کردیم قرار می‌دهیم. و بعد از همه این‌ها، یک سری ثابت‌ها را ست می‌کنیم.
- `forward`: از این متد در حالت آموزش استفاده می‌شود. در ابتدا با استفاده از Feature Extractor ای که قبلاً تعریف کردیم، Feature map ها را بدست آورده و سپس anchor box های مربوط به آن‌ها را می‌سازیم و سپس positive and negative anchor box ها را بدست می‌آوریم. برای این کار از ProposalModule ای که تعریف کردیم استفاده می‌کنیم. سپس با توجه به anchor ها و `reg_offset` ها خطا را برای RPN با توجه به تابع تعریف شده برای آن محاسبه می‌کنیم. برای محاسبه خطا در RPN، برای قسمت طبقه بندی از `binary cross entropy with logits` استفاده می‌کنیم. همچنین برای بخش رگرشن، از `smooth_l1_loss` استفاده می‌کنیم. این تابع به صورت زیر تعریف می‌شود:

For a batch of size N , the unreduced loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^T$$

with

$$l_n = \begin{cases} 0.5(x_n - y_n)^2 / \text{beta}, & \text{if } |x_n - y_n| < \text{beta} \\ |x_n - y_n| - 0.5 * \text{beta}, & \text{otherwise} \end{cases}$$

If reduction is not none, then:

$$\ell(x, y) = \begin{cases} \text{mean}(L), & \text{if reduction} = \text{'mean'}; \\ \text{sum}(L), & \text{if reduction} = \text{'sum'}. \end{cases}$$

سپس خطا نهایی RPN از مجموع خطا بخش classification و regression بدست می‌آید.

- `Inference`: از این متد در حالت استفاده از اعتبارسنجی و یا تست استفاده می‌شود. در اینجا نیز مانند قبل ابتدا feature map ها را بدست می‌آوریم و proposal های پیشنهادی را ارائه داده و از بین آن‌ها با توجه به ترشهولدهای تعریف شده، تعدادی را انتخاب می‌کنیم و آن‌ها را برمی‌گردانیم.

در این بخش به آموزش مدل می‌پردازیم.

در ابتدا یک نمونه (instance) از TwoStageDetector می‌سازیم و سپس پارامترهای آن را ست می‌کنیم. حال با استفاده از تابع‌های داده شده در فایل Hint، سعی می‌کنیم مدل را آموزش دهیم. در مورد خطای RPN گفتیم که برابر خطای بدست آمده از رگرشن و خطای طبقه بندی RPN است. حال در مورد خطا کل مدل صحبت می‌کنیم. خطا این بخش طبیعتاً باید شامل خطا بدست آمده از RPN و خطا دو هد دیگر در انتها مدل باشد که وظیفه خروجی را دارد. بنابراین خطا مدل برابر مجموع خطا این دو head و خطا بدست آمده از RPN خواهد بود. لازم به ذکر است که می‌توان برای خطا هر بخش ضربی در نظر گرفت تا در این صورت به آن بخش توجه بیشتری و یا کم‌تری شود.