|  |  |  |
| --- | --- | --- |
| **پرسش 2** | **نام و نام خانوادگی** | علی صفری |
| **شماره دانشجویی** | 810202153 |
|  | **مهلت ارسال پاسخ** | **۱۴۰3.۰9.13** |

|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس شبکه‌های عصبی و یادگیری عمیق**  **تمرین سوم** | | |

**فهرست**

[**مقدمه** 1](#_Toc191315243)

[**پرسش۲–تشخیص تابلوهای راهنمایی و رانندگی** 2](#_Toc191315244)

[۱-۲.آماده سازی مجموعه داده 2](#_Toc191315245)

[2-۲.تنظیم دقیق و ارزیابی مدل تشخیص شی دو مرحله ای 8](#_Toc191315246)

[1-2-2. تعریف شبکه Faster R-CNN و Resnet-50 به عنوان Backbone 8](#_Toc191315247)

[2-2-2. آماده‌سازی اولیه برای تنظیم دقیق 9](#_Toc191315248)

[3-2-2. عملیات آماده‌سازی و نرمال کردن 12](#_Toc191315249)

[4-2-2. معیارهای IoU و mAP 17](#_Toc191315250)

[5-2-2. بهینه‌ساز و تابع هزینه 20](#_Toc191315251)

[6-2-2. ارزیابی مدل 24](#_Toc191315252)

[7-2-2. ترسیم نمودار AP به ازای مقادیر متفاوت IoU 25](#_Toc191315253)

[8-2-2. ارزیابی مدل بر اساس اندازه 28](#_Toc191315254)

[9-2-2. نمایش نمونه تصویر 34](#_Toc191315255)

[3-۲.تنظیم دقیق و ارزیابی مدل تشخیص شی تک مرحله ای 39](#_Toc191315256)

[1-3-2. تعریف شبکه SSD300 و VGG16 به عنوان Backbone 39](#_Toc191315257)

[2-3-2. آماده‌سازی اولیه برای تنظیم دقیق 42](#_Toc191315258)

[3-3-2. عملیات آماده‌سازی و نرمال کردن 45](#_Toc191315259)

[4-3-2. معیارهای IoU و mAP 47](#_Toc191315260)

[5-3-2. بهینه‌ساز و تابع هزینه 47](#_Toc191315261)

[6-3-2. ارزیابی مدل 50](#_Toc191315262)

[7-3-2. ترسیم نمودار AP به ازای مقادیر متفاوت IoU 51](#_Toc191315263)

[8-3-2. ارزیابی مدل بر اساس اندازه 54](#_Toc191315264)

[9-3-2. نمایش نمونه تصویر 57](#_Toc191315265)

[4-۲. ارزیابی نتایج و مقایسه مدل‌ها 58](#_Toc191315266)

[2-4-1. مقایسه مدل‌های Faster R-CNN و SSD بر اساس آستانه‌های IoUو برای کلاس‌های متخلف 58](#_Toc191315267)

[2-4-2. مقایسه مدل‌های Faster R-CNN و SSD300 بر اساس اندازه اشیاء 63](#_Toc191315268)

[2-4-3. عوامل مؤثر بر عملکرد مدل‌ها 66](#_Toc191315269)

**شکل‌ها**

شکل 10- نمایش سه عکس از مجموعه داده با annotation 3

شکل 11- تابع تشخیص سایز عکس 4

شکل 12 - نام و شماره کلاس‌های موجود 5

شکل 13 - تابع تشخیص کلاس عکس با توجه به class\_id 6

شکل 14 - توزیع فراوانی تصاویر از نظر اندازه و کتگوری 6

شکل 15 - تابع split data و ذخیره اطلاعات و همچنین نمایش distribution 7

شکل 16 - فراوانی مجموعه train و test 8

شکل 17 - تعریف کلاس دیتاست و کتگوری‌ها 10

شکل 18 - ساخت اولیه مدل با بارگیری Faster R-CNN و تعریف دوباره تعداد کلاس‌ها با توجه به مسئله 10

شکل 19- تعریف تابع transform برای نرمال کردن و augmentation 13

شکل 20- کلاس آماده سازی دیتاست بخش اول 15

شکل 21- کلاس آماده سازی دیتاست بخش دوم 16

شکل 22 - ایجاد دیتاست با استفاده از کلاس GTSDBDataset 16

شکل 23- تعریف DataLoader 17

شکل 24 - پیاده‌سازی IoU 19

شکل 25 - پیاده سازی تابع ارزیابی و mAP 19

شکل 26- پیاده‌سازی تابع هزینه در مدل Faster R-CNN 20

شکل 27- تعریف بهینه‌ساز برای مدل Faster R-CNN 21

شکل 28- تابع train در مدل Faster R-CNN 23

شکل 29 - ارزیابی عملکرد مدل Faster R-CNN در حین آموزش با IoU=0.5 25

شکل 30 - mAP هر کلاس 25

شکل 31- تابع ارزیابی مدل با threshold های مختلف 26

شکل 32 - نمودار مقادیر مختلف mAP برای هر کلاس با توجه به threshold انتخابی 27

شکل 33- پیاده‌سازی تابع evaluate by size بخش اول 29

شکل 34 - ارزیابی مدل Faster R-CNN بر اساس سایز 32

شکل 35 - اجرای تابع نمایش تصاویر و مقایسه پیشبینی مدل و واقعیت برای یک نمونه تصویر 36

شکل 36- تابع نمایش تصویر و مقایسه بخش اول 37

شکل 37 - تابع نمایش تصویر و مقایسه بخش دوم 38

شکل 38 - نمونه تصویر مقایسه مدل Faster R-CNN با ground truth 39

شکل 39- آماده‌سازی مدل برای مدل SSD 42

شکل 40 - نرمال‌سازی و آماده سازی تصاویر برای آموزش 46

شکل 41- ارزیابی عملکرد مدل SSD300 در حین آموزش با IoU=0.5 50

شکل 42- mAP هر کلاس 50

شکل 43- نمودار مقادیر مختلف mAP برای هر کلاس با توجه به threshold انتخابی 51

شکل 44- ارزیابی مدل SSD300 بر اساس سایز 55

شکل 45- نمونه تصویر مقایسه مدل SSD300 با ground truth 58

**جدول‌ها**

[جدول 1- جدول mAP با معیار IoU = 0.5 برای هر کلاس و به صورت میانگین 25](#_Toc191315306)

[جدول 2 - ارزیابی مدل Faster R-CNN با threshold های متفاوت 28](#_Toc191315307)

[جدول 3- ارزیابی مدل Faster R-CNN بر اساس سایز 32](#_Toc191315308)

[جدول 4- جدول mAP با معیار IoU = 0.5 برای هر کلاس و به صورت میانگین 51](#_Toc191315309)

[جدول 5 - ارزیابی مدل SSD300 با threshold های متفاوت 54](#_Toc191315310)

[جدول 6- ارزیابی مدل SSD300 بر اساس سایز 55](#_Toc191315311)

# **مقدمه**

برای پیاده‌سازی پروژه از بستر Google Colab به منظور کد نویسی و اجرا استفاده شده است. تمامی مراحل کد و اجرای آن در این گزارش به تفصیل شرح داده شده است.

کد های نوشته شده همگی در پوشه‌ی Code و با پسوند .ipynd ذخیره شده است.

# **پرسش۲–تشخیص تابلوهای راهنمایی و رانندگی**

ابتدا دیتاست خواسته شده از لینک مربوطه دانلود شد و بعد از extract کردن در گوگل درایو ذخیره شد. در ادامه حل این سوال برای دسترسی به دیتاست به آدرسی از گوگل درایو خود اشاره خواهد شد. برای حل این سوال از کتابخانه Pytorch استفاده شده است اما علاوه بر آن در بخش‎های متفاوت بسته به نیاز کتابخانه‎های دیگری مانند torchmetrics برای محاسبه mAP استفاده شده است. از دیگر کتابخانه‌های مورد استفاده torchvision و matplotlib می‌باشد.

کد مربوط به بخش اول سوال در فایل HW3\_2\_1 ذخیره شده است. کد مربوط به بخش دوم و مدل Faster R-CNN در فایل HW3\_2\_1 و کد مربوط به بخش سوم و مدل SSD300 در فایل HW3\_2\_3 و همگی این فایل ها در فولدر Code و داخل Q2 ذخیره شده اند.

**نکات حائز اهمیت در اجرای کدهای این سوال:**

1. تمامی عکس‌های دیتاست دانلود و اکسترکت شده در یک فولدر در گوگل درایو. مسیری که تمامی تصاویر و فایل‌های ذخیره شده در آن قرار گرفته اند : "/content/drive/MyDrive/GTSDB/FullIJCNN2013/".

پس برای اجرای صحیح کد باید تمامی تصاویر و محتویات فایل دانلود شده در مسیر فوق در گوگل درایو در دسترس باشند.

1. برای اجرای صحیح کدها ابتدا باید کد موجود در فایل HW3\_2\_1 اجرا شود و سپس به سراغ کدهای دو بخش بعد بروید چرا که با اجرای فایل مذکور دو فایل با عنوان npy در همان مسیر قبل ذخیره خواهد شد که نتیجه تقسیم کل دیتاست به دو مجموعه آموزش و ارزیابی است. لذا بدون اجرای کد اول این دو فایل در گوگل درایو شما ایجاد نخواهد شد و نتیجتا اجرای کد بخش‌های بعد که مستلزم وجود دو فایل آموزش و تست با پسوند npy هستند با خطا مواجه خواهد شد.

## ۱-۲.**آماده سازی مجموعه داده**

این مجموعه داده حاوی 900 عکس طبیعی از حالت ترافیکی در آلمان است که با فرمت ppm ذخیره شده است. هر عکس حاوی مشخصه (annotation) هایی از جمله شماره کلاس و همچنین مختصات region of interest در عکس است. این مشخصه‌ها برای هر عکس در فایلی تحت عنوان ground truth که در gt.txt ذخیره شده است قرار دارد. هر سطر از این فایل تکست حاوی شماره عکس، چهار عدد که بیان کننده بازه region of interest یا اینجا همان bounding box ما هستند و در نهایت یک عدد به عنوان کلاس آن عکس گزارش شده است. البته لازم به ذکر است که یک عکس می‌تواند حاوی چند تابلو و به عبارتی چندین کلاس و ROI باشد. در این صورت یک شماره از عکس در فایل gt.txt در چند سطر تکرار شده است و در هر سطر bounding box مربوط به آن تابلو خاص و همچنین کلاس آن تابلو به عنوان کلاس گزارش شده است. مجموعه داده حاوی 42 عدد به عنوان کلاس‌های متفاوت است اما به صورت کلی ما چهار دسته بندی برای تابلو ها داریم که شامل prohibitory و danger و mendatory و other می‌باشد. تقسیم بندی اینکه هر شماره کلاس متعلق به کدام کلاس است در فایل gt.txt آمده است. همچنین عکس‌ها دارای ابعاد متفاوت small و medium و large هستند و نحوه تقسیم بندی آنها بر اساس مقاله به این صورت است که عکس‌های کوچک‌تر از 32 پیکسل در رده small بزرگتر از 45 پیکسل large و بین این دو در رده medium قرار می‌گیرد.

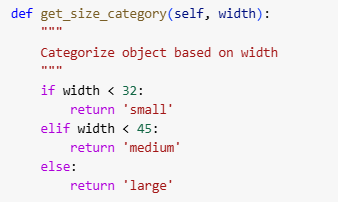
در ادامه به نمایش سه عکس اول می‌پردازیم. هر تابلو با کادر قرمز نمایش داده شده و کلاس مربوطه به رنگ سفید نمایش داده شده است.

A street with cars and buildings

Description automatically generated

شکل 1- نمایش سه عکس از مجموعه داده با annotation

در گام بعد مدل را بر اساس مقاله و بزرگی به سه دسته small که تصاویر کوچک تر از 32 پیکسل بودند و medium بین 32 تا 45 پیکسل و large، بزرگتر از 45 پیکسل تقسیم کردیم. همچنین نوع دیگری از تقسیم بندی تصاویر بر اساس کلاس آنها بوده است که به چهار کلاس prohibitory و danger و mandatory و other تقسیم شده اند. برای تشخیص اینکه هر شماره کلاس متعلق به کدام یک از این چهار کلاس است از فایل ground truth استفاده شده است.

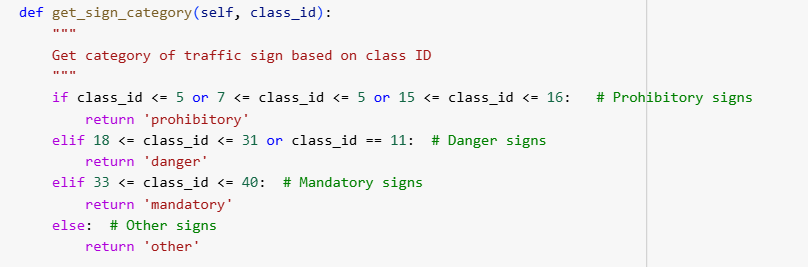


شکل 2- تابع تشخیص سایز عکس

A screenshot of a computer program

Description automatically generated

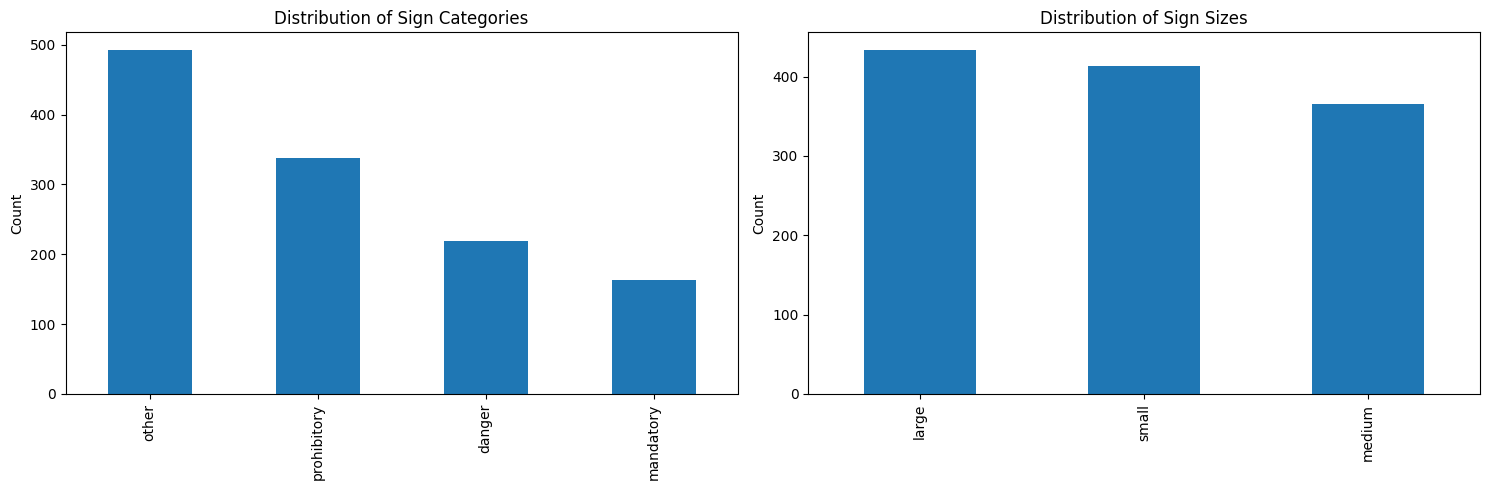
شکل 3 - نام و شماره کلاس‌های موجود



شکل 4 - تابع تشخیص کلاس عکس با توجه به class\_id

نکته مهم در اختصاص هر کلاس ایدی به کلاس مشخصی است که تمامی اعداد از فایل ground truth خوانده شده و در این قسمت بر همین اساس بازه‌ها تعریف شده اند.

در ادامه تابعی را نوشته و با توجه به اینکه هر عکس در چه کلاسی از نظر سایز و دسته بندی تابلو قرار دارد نمودار فراوانی را ترسیم کرده ایم.



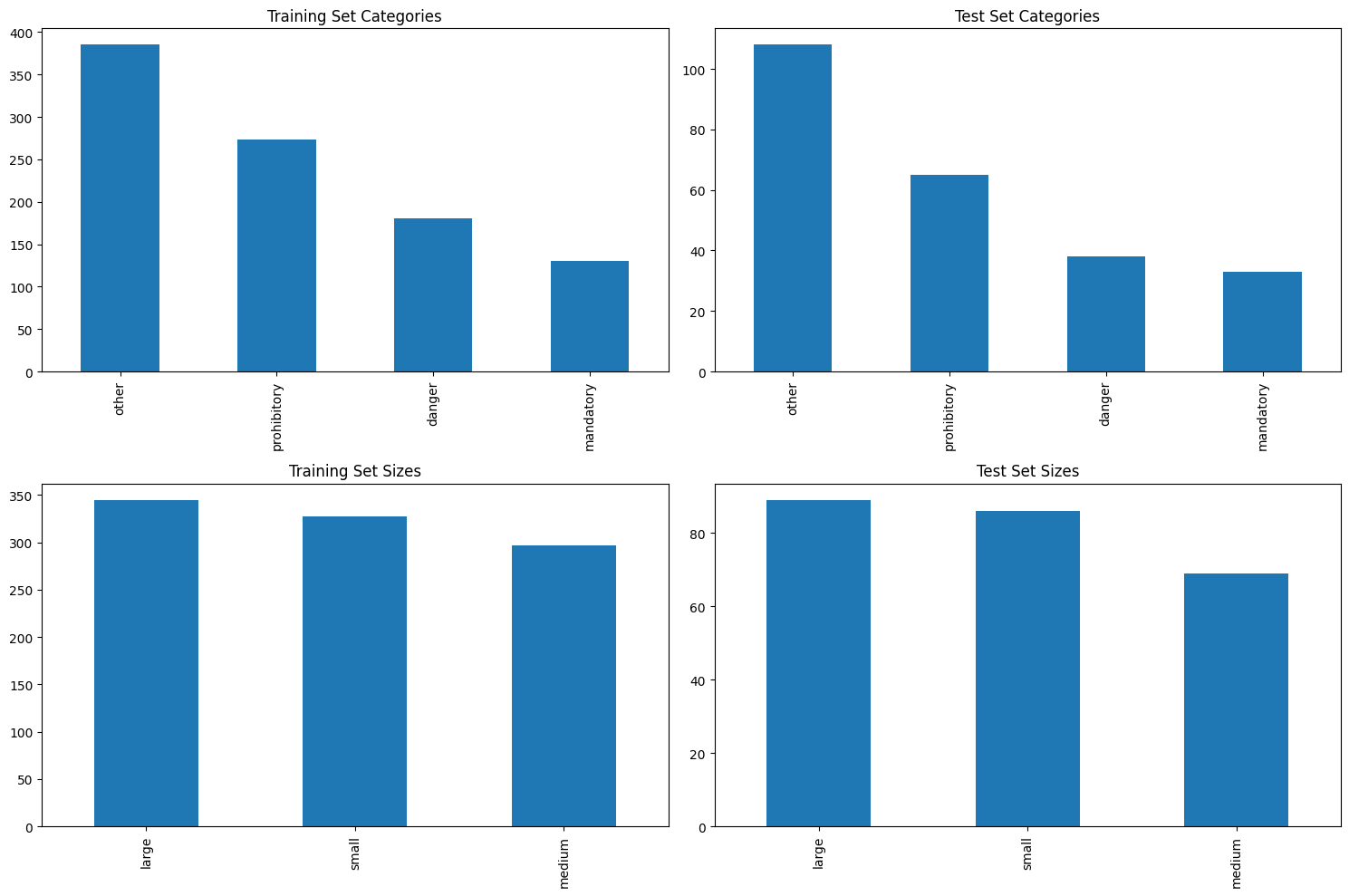
شکل 5 - توزیع فراوانی تصاویر از نظر اندازه و کتگوری

در مرحله بعد 20 درصد داده‌ها را برای test و باقی را برای train اختصاص دادیم. برای این کار تابعی نوشته و بعد از تصادفی کردن ساختار کل تصاویر با استفاده از shuffle به تقسیم دیتا به این دو دسته کردیم. در ادامه کل دیتاست مربوطه را در فایلی تحت عنوان train\_files و test\_files با پسوند npy در گوگل درایو خود ذخیره کردم تا به این دیتا دسترسی داشته و در ادامه نیز از همین تصاویر برای آموزش و ارزیابی استفاده کنیم. پسوند npy نوعی فایل با فرمت باینری و ساخته شده با کتابخانه numpy است که حجم اندکی داشته و اطلاعات لازم که شامل اسم تصاویر ذخیره شده هستند را در خود نگه می‌دارد. در ادامه هر کجا که نیاز به تصاویر داشته باشیم تصاویری که در اینجا ذخیره شده اند را به عنوان تصاویر آموزش استفاده خواهیم کرد و هر کجا نیاز به annotation های مربوط به آن باشد از فایل ground truth استفاده خواهیم کرد. با اینکار به جای صرف فضای اضافی برای ذخیره خود تصاویر در یک فولدر جدا صرفا نام تصاویر را ذخیره کرده ایم.



شکل 6 - تابع split data و ذخیره اطلاعات و همچنین نمایش distribution

در داخل این تابع، تابعی برای نمایش فراوانی دو دسته هم نوشته شده است. فراوانی کلاس آموزش و ارزیابی پس اس تقسیم بندی به فرم زیر است.



شکل 7 - فراوانی مجموعه train و test

## 2-۲.**تنظیم دقیق و ارزیابی مدل تشخیص شی دو مرحله ای**

### 1-2-2. تعریف شبکه Faster R-CNN و Resnet-50 به عنوان Backbone

**Faster R-CNN** (Region-based Convolutional Neural Network) یک چارچوب دو مرحله‌ای برای تشخیص اشیا است که به دلیل دقت بالا به طور گسترده مورد استفاده قرار می‌گیرد. این روش ابتدا از طریق یک **Region Proposal Network (RPN)** به تولید **Region Proposal**‌ها پرداخته و سپس این نواحی پیشنهادی را طبقه‌بندی کرده و جعبه‌های مرزی آن‌ها را اصلاح می‌کند. این رویکرد "coarse-to-fine" امکان موقعیت‌یابی دقیق و شناسایی اشیا را در تصویر فراهم می‌کند. Faster R-CNN به‌ویژه در سناریوهایی که نیاز به دقت بالا دارند، موثر است، اگرچه از نظر سرعت نسبت به روش‌های یک مرحله‌ای مانند SSD یا YOLO کندتر است.

در این پروژه، Faster R-CNN با **ResNet-50**، که یک **Feature Extractor** قوی و کارآمد است، ترکیب شده است. **ResNet-50** (مخفف Residual Network با 50 لایه) از یک چارچوب **Residual Learning** استفاده می‌کند که مشکل **vanishing gradient** در شبکه‌های عمیق را با معرفی **skip connections** کاهش می‌دهد. این اتصالات امکان یادگیری نقشه‌های باقیمانده (residual mappings) را به‌جای نقشه‌های مستقیم فراهم می‌کنند، و این موضوع باعث می‌شود که شبکه حتی در عمق زیاد، ویژگی‌های بهتر و دقیق‌تری یاد بگیرد.

**Backbone ResNet-50** نقش حیاتی در استخراج ویژگی‌های سلسله‌مراتبی و غنی از متن از تصاویر دارد. این روش اطمینان می‌دهد که جزئیات سطح پایین (مانند لبه‌ها) و انتزاع‌های سطح بالا (مانند بخش‌های اشیا) به‌طور موثری ثبت می‌شوند. تعادل بین کارایی محاسباتی و دقت آن، ResNet-50 را به یک انتخاب محبوب برای وظایف تشخیص اشیا، به‌ویژه زمانی که منابع محاسباتی محدود است، تبدیل کرده است. نکته مهم دیگر این است که ResNet-50 روی ImageNet، Pre-train شده است که یک دیتاست بسیار بزرگ است.

در این پروژه، ResNet-50 باعث می‌شود Faster R-CNN بتواند تشخیص دقیق علائم راهنمایی و رانندگی را انجام دهد، با بهره‌گیری از قابلیت آن در یادگیری ویژگی‌های دقیق در مقیاس‌های مختلف اشیا.

### 2-2-2. آماده‌سازی اولیه برای تنظیم دقیق

1. **بارگیری مدل از پیش‌آموزش دیده**

در ابتدا، مدل **Faster R-CNN** با Backbone از نوع **ResNet-50** که از پیش بر روی مجموعه داده **ImageNet** آموزش دیده است، بارگیری می‌شود. این مدل به عنوان پایه برای استخراج ویژگی استفاده می‌شود، و به دلیل پیش‌آموزش آن روی داده‌های عمومی، ویژگی‌های کلی تصویر را به خوبی شناسایی می‌کند.

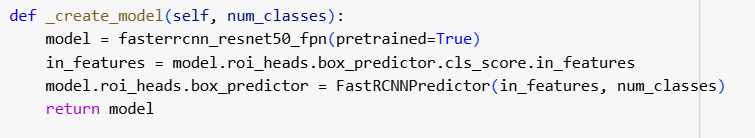
1. **جایگزینی لایه‌های پایانی مدل**

برای تطبیق مدل با مسئله تشخیص علائم راهنمایی و رانندگی، لایه‌های پایانی **Classifier** جایگزین می‌شوند. این لایه‌ها تعداد خروجی‌های مربوط به تعداد کلاس‌ها را تولید می‌کنند که شامل **5 کلاس** است. چهار کلاس اول همان کتگوری‌هایی هستند که در قسمت اول سوال به آن اشاره کردیم: Prohibitory و Danger و Mandatory و Other و یک کلاس هم به عنوان Background. نکته بسیار مهم و حائز اهمیت در این بخش این است که کلاس background در Faster R-CNN به عنوان کلاس صفر از قبل رزرو شده است. و این بدان معنا است که هر کلاس دیگری که بخواهیم جایگزین کنیم باید از یک شروع شود.

A screen shot of a computer program

Description automatically generated

شکل 8 - تعریف کلاس دیتاست و کتگوری‌ها



شکل 9 - ساخت اولیه مدل با بارگیری Faster R-CNN و تعریف دوباره تعداد کلاس‌ها با توجه به مسئله

خط اول کار مورد اول را انجام می‌دهد و مدل را بارگیری می‌کند.

اما خط دوم تعداد **ویژگی‌های ورودی** (Input Features) را که برای لایه cls\_score در **box predictor** مدل مورد نیاز است، استخراج می‌کند.

**شرح دقیق**

* model.roi\_heads:
  + این بخش مربوط به **Region of Interest (RoI) heads** در مدل Faster R-CNN است. وظیفه RoI heads، دو بخش زیر است:
    1. **طبقه‌بندی** (Classification) اشیاء در مناطق پیشنهادی (Regions of Interest) که توسط RPN (Region Proposal Network) تولید شده‌اند.
    2. **پیش‌بینی جعبه‌های مرزی** (Bounding Box Regression) برای آن اشیاء.
* .box\_predictor:
  + این بخش در RoI heads شامل لایه‌های نهایی است که:
    1. نمره هر کلاس را پیش‌بینی می‌کند (cls\_score).
    2. مختصات جعبه‌های مرزی را اصلاح می‌کند (bbox\_pred).
* .cls\_score:
  + این لایه **fully connected (FC)** است که نمرات مربوط به احتمال هر کلاس از جمله background را پیش‌بینی می‌کند.
* .in\_features:
  + این ویژگی تعداد ورودی‌هایی که لایه cls\_score نیاز دارد را مشخص می‌کند. این ورودی‌ها از ویژگی‌هایی که در مراحل قبلی مانند Backbone و RoI Pooling استخراج شده‌اند، تشکیل می‌شوند.

در نتیجه متغیر in\_features اکنون تعداد ویژگی‌های ورودی مورد نیاز برای لایه طبقه‌بندی را نگه می‌دارد. این مقدار در خط بعدی برای ساخت یک **box predictor** جدید استفاده می‌شود.

اما خط سوم **box predictor** اصلی مدل Faster R-CNN را با یک **box predictor** جدید جایگزین می‌کند که تعداد کلاس‌های آن برابر با کلاس‌های موجود در داده‌های پروژه است.

**شرح دقیق**

* FastRCNNPredictor:
  + این کلاس در کتابخانه torchvision تعریف شده و برای جایگزینی آسان **box predictor** در Faster R-CNN استفاده می‌شود.
  + این کلاس شامل دو بخش است:
    1. **classification head**: پیش‌بینی احتمال هر کلاس (به تعداد num\_classes، شامل background).
    2. **bounding box regression head**: پیش‌بینی اصلاحات مختصات جعبه‌های مرزی.
* in\_features:
  + تعداد ویژگی‌های ورودی که از خط اول به دست آمده و به طبقه‌بندی‌کننده جدید منتقل می‌شود.
* num\_classes:
  + تعداد کل کلاس‌ها در مجموعه داده. این مقدار شامل یک کلاس اضافه برای **background** است (برای مناطقی که هیچ شیئی ندارند).

**چگونگی عملکرد**

1. **Box Predictor قدیمی**:
   * مدل اصلی Faster R-CNN که از پیش‌آموزش دیده است (برای مثال روی COCO یا ImageNet)، یک **box predictor** با تعداد کلاس‌های عمومی دارد. مثلاً در COCO، تعداد کلاس‌ها 91 است. 80 کلاس اشیاء + 1 کلاس background + 10 کلاس اضافی
2. **Box Predictor جدید**:
   * این خط، **box predictor** قدیمی را با یک مورد جدید جایگزین می‌کند که تعداد کلاس‌ها را برای داده‌های پروژه تنظیم می‌کند. برای مثال، اگر داده‌های شما شامل سه کلاس "prohibitory"، "danger" و "other" و"mandatory" و background می‌باشد، مقدار num\_classes برابر 5 خواهد بود.

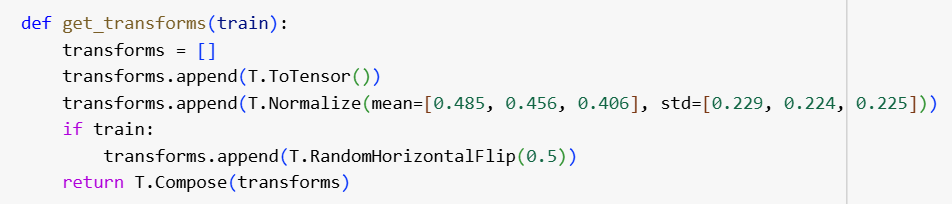
**نتیجه**

* box predictor جدید اکنون می‌تواند جعبه‌های مرزی و کلاس‌ها را برای تعداد دقیق کلاس‌های مورد نیاز در پروژه پیش‌بینی کند.

باقی عملیات‌های آماده سازی و پیاده‌سازی مدل که شامل آموزش آن هم می‌شود در ادامه و بسته به صورت سوال شرح داده شده است.

### 3-2-2. عملیات آماده‌سازی و نرمال کردن

عملیات آماده‌سازی تصاویر برای مدل Faster R-CNN شامل مشخص کردن annotation هر عکس با توجه به فایل gt است. این مراحل شامل مشخص کردن bounding box و label هر عکس است. در گام بعد به سراغ normal کردن عکس می‌رویم. برای این کار مطابق عکس زیر عمل می‌کنیم.



شکل 10- تعریف تابع transform برای نرمال کردن و augmentation

اعداد موجود در دستور Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) میانگین (Mean) و انحراف معیار (Standard Deviation یا Std) رنگ‌های قرمز (R)، سبز (G) و آبی (B) در تصاویر دیتاست ImageNet هستند.

**چرا این اعداد استفاده می‌شود؟**

1. **میانگین (Mean)** : این اعداد نشان‌دهنده میزان روشنایی میانگین هر کانال رنگی هستند. برای مثال، عدد 0.485 مربوط به کانال قرمز است و نشان می‌دهد که روشنایی این کانال به طور میانگین برابر با 0.485 است.
2. **انحراف معیار (Std)** : این اعداد نشان‌دهنده پراکندگی یا پخش‌شدگی شدت رنگ‌ها در تصاویر هستند. به عنوان مثال، مقدار 0.229 برای کانال قرمز به این معنا است که شدت روشنایی در این کانال به اندازه این مقدار بالا و پایین می‌شود.

**کاربرد نرمال‌سازی:**

1. **استاندارد کردن داده‌ها:** نرمال‌سازی باعث می‌شود داده‌های ورودی به شبکه عصبی به یک بازه استاندارد تبدیل شوند. این کار برای بهبود عملکرد مدل بسیار ضروری است. فرمول نرمال‌سازی برای هر کانال به شکل زیر است:

Pixel\_normalized = (Pixel\_original - Mean) / Std

در این فرمول:

* + Pixel\_original مقدار اصلی هر پیکسل (بین 0 و 1) است.
  + Mean میانگین مربوط به کانال رنگی است (برای مثال، 0.485 برای قرمز).
  + Std انحراف معیار کانال رنگی است (برای مثال، 0.229 برای قرمز).

1. **هماهنگی با مدل‌های پیش‌آموزش‌دیده:** مدل‌هایی که روی دیتاست ImageNet آموزش دیده‌اند، انتظار دارند داده‌های ورودی دارای ویژگی‌های آماری مشابه (میانگین و انحراف معیار) باشند. اگر داده‌ها به این صورت نرمال‌سازی نشوند، ممکن است دقت مدل کاهش یابد.

همانطور که در تصویر فوق قابل مشاهده است علاوه برا عملیات نرمال کردن عکس‌ها برای پیش پردازش عملیات داده‌افزایی کوچکی هم به صورت اضافه انجام شده است. این کار الزامی نبوده و صرفا به جهت افزایش دقت مدل به صورت dynamic و random در میان epoch ها چندین عکس به صورت افقی flip شده است. به صورت کلی این کار باعث می‌شود مدل الگوهای موجود را بهتر یاد گرفته و از overfit شدن دور شود. البته لازم به ذکر است که اگر دیتا در مجموعه train ما بوده باشد این کار صورت می‌گیرد و برای داده‌ی تست اینکار انجام نمی‎شود.

در نهایت با استفاده از T.Compose تنسور تولید شده ما قابل فراخوانی می‌شود. حال دیتا آماده آموزش است.

برای مهیا کردن دیتاست برای آموزش و ارزیابی کلاسی تحت عنوان GTSDBDataset تعریف شده است. در این کلاس که تا پیش از این چندین تابع به کار رفته در آن را هم دیده‌اید مراحل آماده سازی دیتاست برای آموزش انجام می‌شود. این مراحل به شرح زیر است:

* 1. ابتدا annotation های هر عکس مشخص شده است. به این معنا که از فایل gt، Bounding box مشخص شده و شماره کلاس اون مشخص شده است.
  2. در گام بعد با توجه به شماره کلاس هر عکس مشخص شده که در کدام یک از چهار کلاس prohibitory و danger و mandatory و other تعلق دارد.
  3. در گام بعد یک دیکشنری برای هر عکس درست شده است که در اون box مربوطه و label اون عکس که به هر کلاس یک عدد از یک تا چهار داده شده نسبت داده شده و در نهایت image\_id که اسم اون عکس هست.
  4. در نهایت عکس از تابع get\_transform عبور می‌کند تا عملیات نرمال کردن و در صورتی که داده آموزش باشد، عملیات داده افزایی هم به صورت رندوم روی آن اعمال شود.

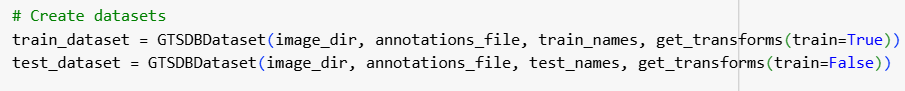


شکل 11- کلاس آماده سازی دیتاست بخش اول

A screenshot of a computer program

Description automatically generated

شکل 12- کلاس آماده سازی دیتاست بخش دوم



شکل 13 - ایجاد دیتاست با استفاده از کلاس GTSDBDataset

بعد از آماده شدن دیتاست برای آموزش و ارزیابی از DataLoader استفاده کردیم تا سرعت و استفاده از GPU را بهینه تر کنیم.

A screenshot of a computer code

Description automatically generated

شکل 14- تعریف DataLoader

در تعریف دیتالودر ابتدا دیتاست مربوط به هر بخش فرستاده شده است سپس batch size مساوی 8 در نظر گرفته شده، سپس shuffle فقط برای آموزش روشن شده و در بخش بعد چگونگی ارسال عکس ها در دیتالودر مشخص شده که از زیپ استفاده شده، به این معنا که هر عکس و annotation مربوط بهش ارسال شده است. در انتها تعداد worker ها برابر دو قرار داده شده که به معنای **worker processes** است که به صورت همزمان به کار گرفته شده اند. این کار به افزایش سرعت کمک می‌کند.

همه چیز آماده و مهیا برا آموزش است. برای آموزش کلاس TrafficSignDetector نوشته شده است. و در این قسمت بخش مهم کار تابع train است. اما توابع evaluation هم داریم که در ادامه به آنها اشاره می‌کنیم.

**در مرحله آموزش فرض‌هایی صورت گرفته است که به آنها اشاره می‌شود:**

1. **تعداد epoch:** این مقدار در مقاله یا صورت سوال اشاره نشده است و ما با توجه به زمانبر بودن اجرا و دست یافتن به دقت قابل قبول، تعداد 10 ایپاک را در نظر گرفتیم.
2. **Batch size:** این مقدار هم مشخصا تعریف نشده است. برای این سوال عدد 8 را برای batch size در نظر گرفته‌ایم.

اکنون مدل آماده آموزش است و با استفاده از تابع زیر که در کلاس TrafficSignDetector تعریف شده این کار را انجام می‎‌دهیم. برای ارزیابی و چگونگی انجام آن در ادامه توضیح داده شده است.

### 4-2-2. معیارهای IoU و mAP

**IoU (Intersection over Union)**

**IoU** یکی از معیارهای اصلی در ارزیابی دقت پیش‌بینی‌های مدل‌های تشخیص شی است. IoU میزان همپوشانی میان جعبه پیش‌بینی شده (Predicted Bounding Box) و جعبه واقعیت (Ground Truth Bounding Box) را محاسبه می‌کند. فرمول IoU به صورت زیر است:

* **مقدار بالا (مثلاً 0.75):** پیش‌بینی باید دقیقاً با جعبه واقعیت مطابقت داشته باشد.
* **مقدار پایین (مثلاً 0.5):** پیش‌بینی با همپوشانی جزئی نیز پذیرفته می‌شود.

در ارزیابی مدل:

* اگر **IoU > آستانه (مثلاً 0.5)** باشد، پیش‌بینی به عنوان درست (True Positive) پذیرفته می‌شود.
* در غیر این صورت، پیش‌بینی به عنوان خطا (False Positive) محسوب می‌شود.

**mAP (Mean Average Precision)**

**mAP** یک معیار ارزیابی کلی برای مدل‌های تشخیص شی است که هم **دقت (Precision)** و هم **بازخوانی (Recall)** را در بر می‌گیرد. ابتدا، برای هر کلاس **AP (Average Precision)** محاسبه شده و سپس میانگین آن‌ها به عنوان mAP گزارش می‌شود.

فرمول:

*mAP = (1/N) Σ AP\_i*

​ AP = ∫ p(r) dr

* **P(r):** دقت در یک مقدار خاص بازخوانی.
* **mAP:** میانگین AP برای تمامی کلاس‌ها.

mAP با IoU در ارتباط است زیرا محاسبه AP به تعداد پیش‌بینی‌های درست (TPs) که از یک آستانه IoU عبور می‌کنند، بستگی دارد.

**پیاده‌سازی IoU**

در کد، IoU به طور غیرمستقیم برای ارزیابی پیش‌بینی‌ها استفاده شده است. IoU در متریک‌های از پیش تعریف‌شده مثل **MeanAveragePrecision** از کتابخانه PyTorch Metrics پیاده‌سازی شده است:



شکل 15 - پیاده‌سازی IoU

**iou\_type="bbox"** مشخص می‌کند که IoU برای جعبه‌ها محاسبه می‌شود.

این تابع خروجی‌های مدل را با داده‌های واقعیت مقایسه کرده و IoU و سایر معیارها مثل mAP را محاسبه می‌کند.

**پیاده‌سازی mAP**

mAP در کد شما برای ارزیابی دقت مدل در طول فرآیند اعتبارسنجی استفاده می‌شود:

A screenshot of a computer code

Description automatically generated

شکل 16 - پیاده سازی تابع ارزیابی و mAP

در این تابع ابتدا مدل ارزیابی شده است سپس با استفاده از MeanAveragePrecision از مدل خواسته شده است که بر اساس bounding box مقدار IoU را محاسبه کند و در ادامه در قالب یک حلقه label های واقعی و پیشبینی شده آپدیت شده اند و در نهایت با استفاده از compute محاسبات انجام شده و مقدار mAP گزارش شده است. نکته مهم در این بخش این است که در تابع MeanAvaragePrecision فرض بر شروع کلاس‌ها از صفر است اما ما می‌دانیم که در مدل خود از عدد 1 شروع کردیم چرا که کلاس background با عدد صفر از قبل رزرو شده بود. به همین دلیل شماره label منهای یک شده است تا بجای یک تا چهار از صفر تا سه باشد و عملیات را به درستی انجام دهد. در صورت عدم انجام این کار عدد mAP منفی یک گزارش می‌شود که صحیح نمی‌باشد. با اینکار مقدار mAP برای هر کلاس و با توجه به IoU threshold تعریف شده محاسبه می‌شود. نکته حائز اهمیت در این بخش این است که به صورت پیش فرض و با توجه به صورت سوال این threshold برابر با **0.5** در نظر گرفته خواهد شد. اما برای ارزیابی بهتر مدل بهتر است با IoU های متفاوت بررسی کنیم که در بخش‌های بعدی به این مورد خواهیم پرداخت.

### 5-2-2. بهینه‌ساز و تابع هزینه

**تابع هزینه**

برای تابع هزینه یا loss از آنچه در درون خود Faster R-CNN بوده استفاده کرده ایم که به شکل زیر پیاده سازی شده است:

A close up of words

Description automatically generated

شکل 17- پیاده‌سازی تابع هزینه در مدل Faster R-CNN

**خطاها (loss)** در یک دیکشنری ذخیره می‌شوند و شامل چهار جزء هستند:

* **loss\_classifier**: خطای Cross-Entropy برای کلاسه‌بندی.
* **loss\_box\_reg**: خطای Smooth L1 برای تنظیم موقعیت جعبه.
* **loss\_objectness**: خطای Binary Cross-Entropy برای احتمال حضور شیء در RPN.
* **loss\_rpn\_box\_reg**: خطای Smooth L1 برای تنظیم موقعیت جعبه در RPN.

فرمول کلی محاسبه خطا به شکل زیر است:

Total Loss = loss\_classifier + loss\_box\_reg + loss\_objectness + loss\_rpn\_box\_reg

این خطای چند وظیفه‌ای برای فرآیند تشخیص دو مرحله‌ای طراحی شده است:

1. **مرحله اول (RPN)**:
   * خطای احتمال حضور شیء و خطای موقعیت جعبه در RPN شبکه تولید ناحیه را آموزش می‌دهند.
2. **مرحله دوم**:
   * خطای کلاسه‌بندی و تنظیم موقعیت جعبه نتایج نهایی را بهبود می‌بخشند.

* **چرا Smooth L1 Loss انتخاب شده؟**
  + این تابع ترکیبی از مزایای L1 و L2 است:
    - **حساسیت کم به نقاط پرت**: برخلاف L2 که در برابر نقاط پرت (outliers) حساسیت بالایی دارد، Smooth L1 تاثیر این نقاط را کاهش می‌دهد.
    - **آموزش پایدارتر**: زمانی که خطا کوچک است، مانند L2 رفتار می‌کند و زمانی که خطا بزرگ‌تر می‌شود، شبیه L1 رفتار می‌کند.
  + این ویژگی‌ها آن را برای تنظیم موقعیت جعبه‌ها (Bounding Boxes) بسیار مناسب می‌کند، زیرا جعبه‌های اولیه ممکن است فاصله زیادی از مقدار واقعی داشته باشند.
* **چراCross-Entropy انتخاب شده؟**
  + **معیار استاندارد کلاسه‌بندی**: Cross-Entropy به خوبی احتمال توزیع خروجی مدل را با برچسب‌های واقعی مقایسه می‌کند.
  + **قابلیت انطباق برای چندکلاسه**: این تابع به راحتی برای مسائلی با تعداد زیاد کلاس‌ها (مثل شناسایی اشیاء مختلف) قابل استفاده است.
  + **سادگی و اثربخشی**: در مسائلی مانند تشخیص اشیاء، این تابع به سرعت همگرایی مدل را تضمین می‌کند.

**بهینه‌ساز**

برای بهینه‌ساز از AdamW استفاده شده است. همچنین از lr\_scheduler هم استفاده شده است تا امکان کاهش و تغییر lr به شکل بهتری مهیا باشد و عملکرد مدل را بهبود بخشد.



شکل 18- تعریف بهینه‌ساز برای مدل Faster R-CNN

ویژگی‌های AdamW:

* ترکیب نرخ یادگیری تطبیقی Adam با وزن‌زدایی مناسب.
* نرخ یادگیری پایین (0.0001) برای تنظیم دقیق مدل پیش‌آموزش‌دیده. (عدد نرخ یادگیری اولیه این عدد فرض شده است و الزاما جایی اشاره نشده حتما از این عدد شروع شود.)
* وزن‌زدایی (Weight Decay) برابر 0.0005 برای جلوگیری از بیش‌برازش (Overfitting).

**دلایل انتخاب AdamW به عنوان بهینه‌ساز**

**.Adaptive Learning Rate**

* **چرا انتخاب شده؟**
  + AdamW از نرخ یادگیری تطبیقی استفاده می‌کند، که باعث می‌شود مدل برای هر پارامتر نرخ یادگیری بهینه پیدا کند.
  + این ویژگی باعث می‌شود مدل سریع‌تر به جواب بهینه برسد، خصوصاً در مسائل پیچیده با تعداد پارامترهای زیاد.

**Weight Decay بهینه**

* **چرا مهم است؟**
  + در مدل‌هایی مانند Faster R-CNN و SSD که تعداد زیادی پارامتر دارند، جلوگیری از **بیش‌برازش** (Overfitting) ضروری است.
  + AdamW از وزن‌زدایی (Weight Decay) مناسب استفاده می‌کند، که مستقیماً بر روی پارامترهای بهینه‌سازی اعمال می‌شود و باعث تنظیم بهتر شبکه می‌شود.

**ترکیب مومنتوم (Momentum)**

* **چرا مفید است؟**
  + مومنتوم به فرار از کمینه‌های محلی (Local Minima) کمک می‌کند و مدل را به سمت کمینه سراسری (Global Minimum) هدایت می‌کند.
  + این ویژگی خصوصاً در مسائل پیچیده مانند شناسایی اشیاء بسیار مفید است.

**پایداری در آموزش**

* AdamW به دلیل رفتار پایدار در مسائل با داده‌های پیچیده و حجیم مثل COCO و GTSDB، گزینه‌ای محبوب است.
* این پایداری به دلیل ترکیب نرخ یادگیری تطبیقی و Weight Decay موثر است.

اکنون که تمامی موارد مورد نیاز برای پیاده سازی و آموزش مدل توضیح داده و انجام شده است تابع train را پیاده‌سازی کرده‌ایم.



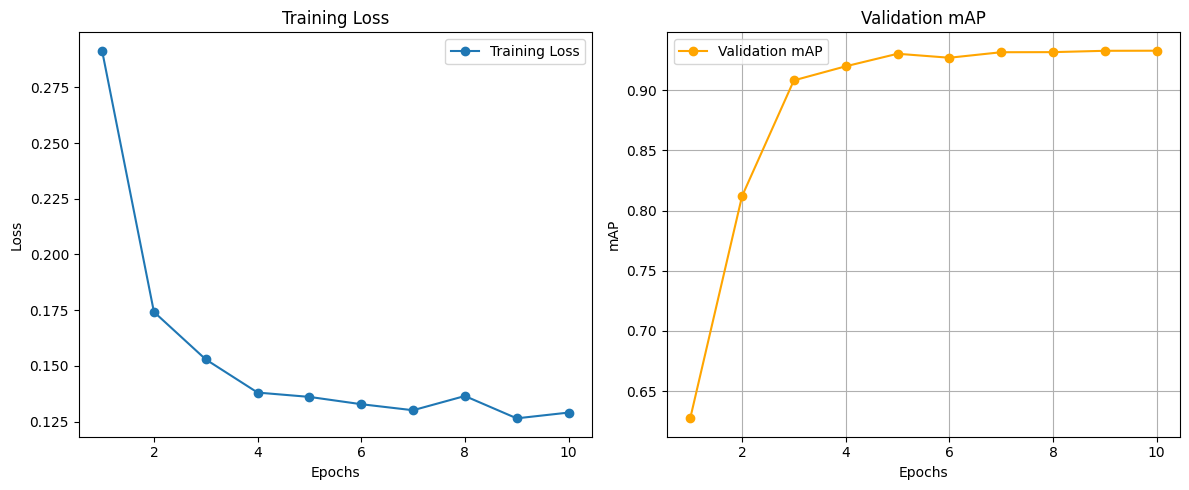
شکل 19- تابع train در مدل Faster R-CNN

از آنجایی که الزامی به توضیح تک تک بخش‌های کد نبوده است نیازی نیست تنها به توضیح دو خط از سه خط اول می‌پردازیم که اهمیت ویژه ای در ساختار کد ما دارد.

1. **params = [p for p in self.model.parameters() if p.requires\_grad]:**
   * این خط پارامترهای مدلی که باید گرادیان آن‌ها محاسبه و به‌روز شود را فیلتر می‌کند.
   * **هدف:**
     + وقتی بخواهید برخی از لایه‌ها (مانند لایه‌های اولیه در یادگیری انتقالی) ثابت بمانند و به‌روزرسانی نشوند، می‌توانید با تنظیم requires\_grad=False این کار را انجام دهید. این کد فقط پارامترهایی که باید به‌روزرسانی شوند را انتخاب می‌کند.
2. **lr\_scheduler = torch.optim.lr\_scheduler.StepLR(optimizer, step\_size=3, gamma=0.1):**
   * این خط یک برنامه‌ریز نرخ یادگیری از نوع StepLR را تعریف می‌کند.
   * **هدف:**
     + نرخ یادگیری (Learning Rate) تأثیر زیادی در آموزش مدل دارد. این تابع هر چند دوره (مثلاً هر ۳ دوره) نرخ یادگیری را به میزان gamma (مثلاً ۰.۱) کاهش می‌دهد. این کاهش تدریجی باعث می‌شود که مدل در ابتدا سریع‌تر یاد بگیرد و در ادامه به آرامی به سمت همگرایی برود.

### 6-2-2. ارزیابی مدل

اکنون تمام پارامترهای مورد نیاز مدل به صورت کامل تنظیم شده و مدل آموزش داده شده است. نتیجه آموزش و بررسی دقت آن بستگی به IoU تنظیم شده دارد. مدل را با استفاده از IoU = 0.5 در طول آموزش مورد ارزیابی قرار داده‌ایم. با این کار روند افزایش mAP با توجه به این threshold به خوبی قابل مشاهده است. برای این کار تابع evaluate نوشته شده است که بعد از هر ایپاک فراخوانی شده و کار ارزیابی با توجه به معیارهای تعیین شده را برای ما انجام می‌دهد. البته این مورد در صورت سوال خواسته نشده است و در این بخش به صورت اضافه انجام می‌گیرد. تصویر تابع evaluate در بخشی که مربوط به پیاده سازی mAP بود با عنوان " پیاده سازی تابع ارزیابی و mAP " آورده شده است. در این تابع ابتدا معیار mAP تعریف شده و سپس با توجه به label واقعی و پیشبینی شده توسط مدل و همینطور IoU تعریف شده که در اینجا برابر 0.5 است پیاده سازی شده است. اما نتیجه ارزیابی مدل در حین آموزش و در نهایت دقت هر کلاس و دقت نهایی در تصاویر زیر آورده شده است.



شکل 20 - ارزیابی عملکرد مدل Faster R-CNN در حین آموزش با IoU=0.5

روند نزولی loss و صعودی mAP نشان از عملکرد صحیح و مناسب مدل است چرا که می‌دانیم مقدار loss در هر مرحله و با عملیات ریاضیتی داخل optimizer باید کاهش یابد و از طرفی مدل در هر گام باید بهتر از گام قبل عمل کند.



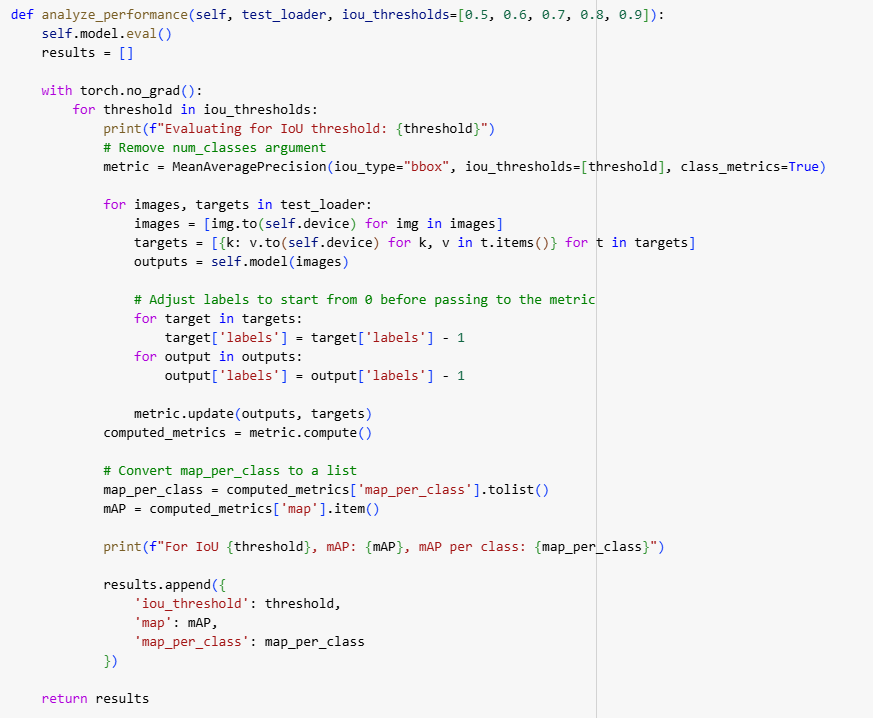
شکل 21 - mAP هر کلاس

جدول 1- جدول mAP با معیار IoU = 0.5 برای هر کلاس و به صورت میانگین

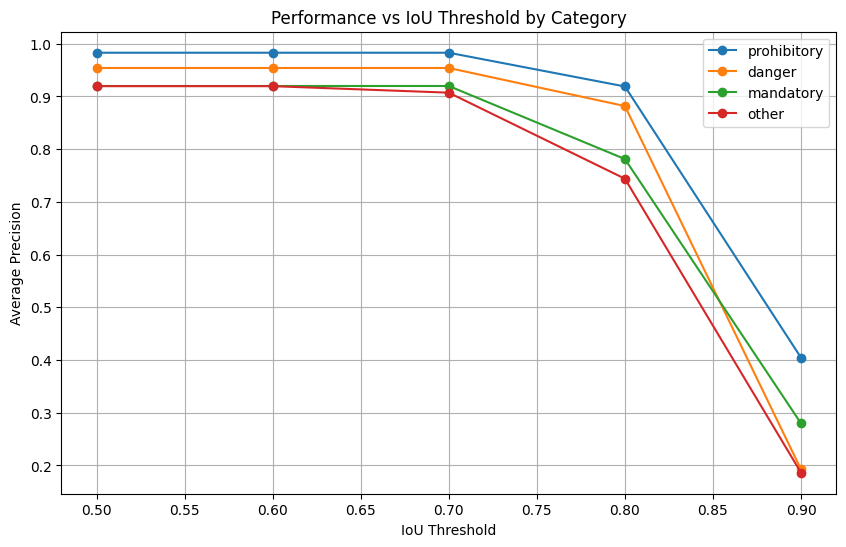
|  |  |
| --- | --- |
|  | mAP |
| **Prohibitory**  **Danger** | 97.24  95.13 |
| **Mandatory** | 88.41 |
| **Other** | 92.39 |
| **میانگین** | **93.29** |

### 7-2-2. ترسیم نمودار AP به ازای مقادیر متفاوت IoU

برای ارزیابی مدل با انواع IoU تابعی مجدد نوشته ایم تا با اختصاص مقادیر مختلف threshold بتوانیم ارزیابی لازم را انجام دهیم.



شکل 22- تابع ارزیابی مدل با threshold های مختلف



شکل 23 - نمودار مقادیر مختلف mAP برای هر کلاس با توجه به threshold انتخابی

این نمودار نشان‌دهنده رابطه بین میانگین دقت (mAP) و آستانه IoU برای دسته‌بندی‌های مختلف (prohibitory، danger، mandatory، other) است. تحلیل نمودار را می‌توان به‌صورت زیر بسط داد:

همان‌طور که در نمودار مشاهده می‌کنیم، مقدار mAP با کاهش آستانه IoU افزایش می‌یابد. این موضوع منطقی است زیرا کاهش آستانه IoU به معنای کاهش سخت‌گیری در ارزیابی مدل است. در واقع، با کاهش مقدار IoU موردنیاز برای درستی پیش‌بینی، مدل می‌تواند تعداد بیشتری از باکس‌های پیش‌بینی‌شده را به‌عنوان True Positive در نظر بگیرد، که باعث افزایش میانگین دقت می‌شود.

در این نمودار، دقت برای تمامی کلاس‌ها در آستانه‌های پایین‌تر IoU بسیار بالا (بالای ۹۰ درصد) است، که نشان‌دهنده عملکرد کلی خوب مدل در تمامی دسته‌ها است. این نکته مثبت به ما می‌گوید که مدل به‌طور کلی توانایی بالایی در تشخیص صحیح دارد.

بااین‌حال، اختلافاتی میان کلاس‌ها دیده می‌شود:

* کلاس‌های **Mandatory** و **Other** دقت کمتری نسبت به **Prohibitory** و **Danger** دارند، به‌ویژه در آستانه‌های IoU بالا. این اختلاف می‌تواند به‌دلایل مختلفی باشد، از جمله تنوع کمتر داده‌های آموزشی برای این کلاس‌ها، شباهت بیشتر این کلاس‌ها به یکدیگر، یا چالش‌های بیشتری در تشخیص این نوع علائم.
* از سوی دیگر، کلاس **Prohibitory** عملکرد بسیار بهتری در تمامی مقادیر IoU دارد، که احتمالاً به‌دلیل تمایز واضح‌تر این کلاس‌ها از دیگر دسته‌ها یا تعداد بیشتر داده‌های مرتبط با این کلاس در مجموعه داده باشد.

این نمودار همچنین نشان می‌دهد که با افزایش آستانه IoU (مثلاً به ۰.۹)، دقت به شدت کاهش می‌یابد. این موضوع طبیعی است زیرا سخت‌گیری مدل در تشخیص درست افزایش می‌یابد و تنها پیش‌بینی‌هایی که به‌طور دقیق با باکس واقعی همپوشانی دارند به‌عنوان درست در نظر گرفته می‌شوند.

**جمع‌بندی:** نمودار نشان می‌دهد که مدل عملکرد بسیار خوبی دارد، اما توجه به اختلاف عملکرد میان کلاس‌ها می‌تواند راهنمایی برای بهبود مدل باشد. با افزایش تعداد داده‌ها یا تنوع آن‌ها برای کلاس‌های Mandatory و Other، می‌توان انتظار داشت که دقت این دسته‌ها نیز بهبود یابد.

جدول 2 - ارزیابی مدل Faster R-CNN با threshold های متفاوت

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | mAP | | | | |
|  | IoU = 0.9 | IoU = 0.8 | IoU = 0.7 | IoU = 0.6 | IoU = 0.5 |
| **Prohibitory** | 30.53 | 89.29 | 97.24 | 97.24 | 97.24 |
| **Danger** | 26.67 | 77.89 | 93.09 | 95.12 | 95.12 |
| **Mandatory** | 24.72 | 82.66 | 91.99 | 88.41 | 88.41 |
| **Other** | 14.20 | 86.07 | 92.39 | 92.39 | 92.39 |
| **میانگین** | **24.03** | **83.98** | **92.78** | **93.29** | **93.23** |

### 8-2-2. ارزیابی مدل بر اساس اندازه

برای ارزیابی مدل بر اساس اندازه کافی است همان معیار mAP را در نظر داشته باشیم و با توجه به سایز تصاویر، mAP محاسبه شده برای تصاویر در سایز های مختلف را محاسبه کرده و میانگین گیری کنیم. به عبارت دیگر تصاویری که در هر کدام از سایزهای small, medium و large هستند را به صورت جداگانه و یک کلاس در نظر می‎‌گیریم و عملکرد و دقت مدل را در ارزیابی تصاویر با آن سایز محاسبه می‌کنیم.

برای اینکار تابعی با عنوان evaluate\_by\_size نوشته شده است.

****

شکل 24- پیاده‌سازی تابع evaluate by size بخش اول

**پارامترها:**

1. **data\_loader**:
   * این ورودی شامل داده‌هایی است که برای ارزیابی مدل استفاده می‌شود.
   * به طور کلی، data\_loader یک مجموعه از تصاویر و باکس‌های محدودکننده مربوط به اشیاء (Ground Truth) است.
2. **size\_thresholds**:
   * یک تاپل که آستانه‌هایی را برای طبقه‌بندی اندازه اشیاء به کوچک، متوسط و بزرگ مشخص می‌کند.
   * مثلاً size\_thresholds=(32, 45) به این معنی است که:
     + اشیاء با عرض کمتر از ۳۲ پیکسل کوچک هستند.
     + اشیاء با عرض بین ۳۲ و ۴۵ پیکسل متوسط هستند.
     + اشیاء با عرض بزرگ‌تر از ۴۵ پیکسل بزرگ محسوب می‌شوند.

**ساختار کد:**

**1.فعال‌سازی حالت ارزیابی مدل:**

* مدل به حالت eval می‌رود. در این حالت، لایه‌هایی مثل Dropout یا BatchNorm به‌صورت ثابت عمل می‌کنند و داده‌های جدیدی یاد نمی‌گیرند.

**2.تعریف دیکشنری برای نگه‌داری متریک‌ها:**

* سه متریک MeanAveragePrecision برای اندازه‌های کوچک، متوسط و بزرگ تعریف شده‌اند.
* iou\_type="bbox" نشان می‌دهد که ارزیابی برای bounding box انجام می‌شود.
* class\_metrics=True به این معنی است که عملکرد برای هر کلاس جداگانه نیز ثبت می‌شود.
* IoU threshold = 0.5 با توجه به صورت سوال.

**3.پردازش داده‌ها در حلقه:**

در این حلقه، هر بار تعدادی تصویر و Ground Truth (targets) از data\_loader گرفته می‌شود.

**انتقال داده‌ها به GPU/CPU:**

* تصاویر و Ground Truth‌ها به دستگاهی که مدل روی آن قرار دارد CPU یا GPU منتقل می‌شوند.

**4. پیش‌بینی مدل:**

* مدل bounding boxها و کلاس‌های پیش‌بینی‌شده را برای تصاویر ورودی برمی‌گرداند.

**5. تنظیم برچسب‌ها:**

* برچسب‌ها یک عدد کم می‌شوند تا از ۰ شروع شوند. این معمولاً به دلیل تفاوت فرمت داده‌های Ground Truth و مدل است.

**6. دسته‌بندی اندازه اشیاء:**

* عرض هر باکس محاسبه می‌شود: width = box[2] - box[0] اختلاف بین مختصات x گوشه‌های چپ و راست.
* اندازه باکس بر اساس size\_thresholds به یکی از دسته‌های small, medium, large اختصاص داده می‌شود.

**7. جداسازی داده‌ها بر اساس اندازه:**

داده‌های Ground Truth (باکس‌ها و برچسب‌ها) بر اساس اندازه جدا می‌شوند. این کار با استفاده از ایندکس‌هایی که مربوط به اندازه خاص هستند، انجام می‌شود.

**8. فیلتر کردن پیش‌بینی‌ها:**

ابتدا IoU (Intersection over Union) بین باکس‌های پیش‌بینی‌شده و Ground Truth محاسبه می‌شود.

* پیش‌بینی‌هایی که IoU آن‌ها بیشتر از ۰ است، به عنوان پیش‌بینی معتبر انتخاب می‌شوند.

**9. به‌روزرسانی متریک:**

داده‌های مربوط به پیش‌بینی‌ها و Ground Truth برای اندازه خاص به متریک مربوطه اضافه می‌شوند.

**10. محاسبه mAP نهایی:**

در انتها، mAP برای هر اندازه محاسبه شده و به دیکشنری size\_map اضافه می‌شود.

**نتیجه خروجی:**

* این تابع یک دیکشنری به نام size\_map برمی‌گرداند که شامل mAP برای سه اندازه small, medium, large است.

A graph of a performance by object size

Description automatically generated

شکل 25 - ارزیابی مدل Faster R-CNN بر اساس سایز

جدول 3- ارزیابی مدل Faster R-CNN بر اساس سایز

|  |  |  |  |
| --- | --- | --- | --- |
|  | mAP for IoU = 0.5 | | |
|  | Large | Medium | Small |
| mAP for IoU = 0.5 | **95.81** | **90.25** | **84.66** |

این نمودار نشان‌دهنده عملکرد مدل در تشخیص اشیاء با اندازه‌های مختلف است. محور افقی سه دسته اندازه اشیاء small، medium، large را نشان می‌دهد و محور عمودی مقدار mAP (میانگین دقت) را برای هر دسته اندازه مشخص می‌کند. تحلیل دقیق نمودار به شرح زیر است:

**مشاهده کلی:**

1. **اشیاء کوچک (small):**
   * مقدار mAP برای این دسته کمترین مقدار را دارد و در حدود **0.8** است.
   * این موضوع نشان می‌دهد که مدل در تشخیص اشیاء کوچک نسبت به دسته‌های دیگر عملکرد ضعیف‌تری دارد.
2. **اشیاء متوسط (medium):**
   * مقدار mAP در این دسته به حدود **0.9** افزایش یافته است.
   * این نشان می‌دهد که مدل در تشخیص اشیاء متوسط عملکرد بهتری نسبت به اشیاء کوچک دارد.
3. **اشیاء بزرگ (large):**
   * مقدار mAP در این دسته تقریباً به **1.0** می‌رسد.
   * این مقدار بالاترین دقت را نشان می‌دهد و بیانگر این است که مدل در شناسایی اشیاء بزرگ عملکرد بسیار خوبی دارد.

**دلایل عملکرد متفاوت در اندازه‌ها:**

1. **عملکرد پایین در اشیاء کوچک:**
   * **مشکل:** اشیاء کوچک معمولاً در تصاویر وضوح کمتری دارند و ممکن است به دلیل اندازه کوچک‌تر، باکس‌های پیش‌بینی‌شده توسط مدل نتوانند به‌خوبی با Ground Truth همپوشانی داشته باشند.
   * **راه‌حل:** افزایش داده‌های آموزشی شامل اشیاء کوچک یا استفاده از مدل‌هایی با رزولوشن ورودی بالاتر می‌تواند عملکرد را بهبود دهد.
2. **عملکرد متوسط در اشیاء متوسط:**
   * مدل توانسته است اشیاء متوسط را با دقت خوبی شناسایی کند. این دسته معمولاً چالش خاصی ندارد، زیرا اندازه اشیاء نه آن‌قدر کوچک است که وضوح کم باشد و نه آن‌قدر بزرگ که با محدودیت‌های دیگر مواجه شویم.
3. **عملکرد بالا در اشیاء بزرگ:**
   * اشیاء بزرگ معمولاً در تصاویر وضوح بالایی دارند و پیش‌بینی باکس‌های محدودکننده برای این اشیاء آسان‌تر است.
   * همچنین، همپوشانی (IoU) بین باکس‌های پیش‌بینی‌شده و Ground Truth برای اشیاء بزرگ بالاتر است، زیرا اختلاف مختصاتی کمتر تأثیرگذار است.

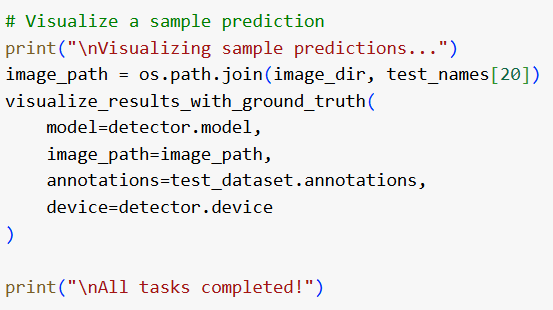
**تحلیل کلی و نتیجه‌گیری:**

* مدل در دسته‌بندی اشیاء با اندازه‌های بزرگ عملکرد فوق‌العاده‌ای دارد.
* با این حال، عملکرد مدل در تشخیص اشیاء کوچک قابل بهبود است. این مسئله ممکن است به دلیل چالش‌های مربوط به وضوح کم، داده‌های ناکافی یا طراحی معماری مدل باشد.
* عملکرد در اشیاء متوسط متعادل است و مدل در این دسته نیاز به تغییرات زیادی ندارد.

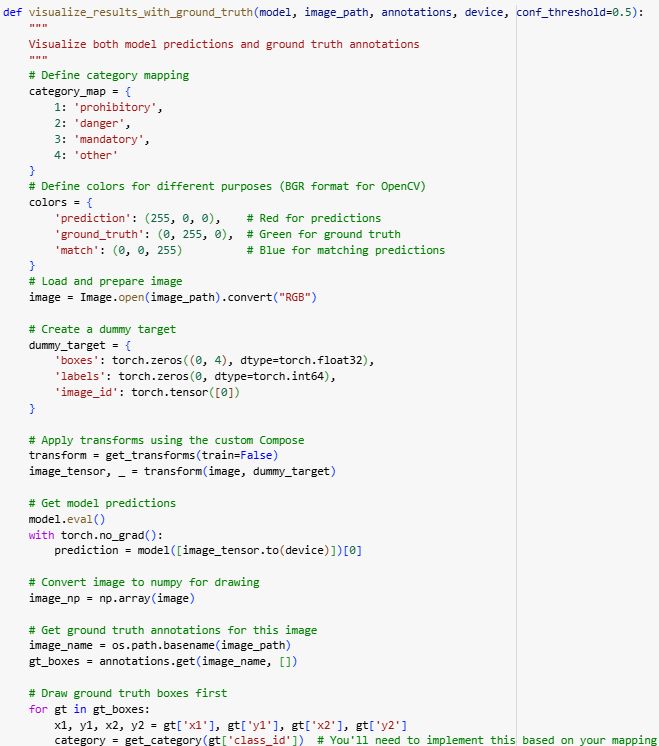
1. **استفاده از معماری مناسب:**
   * مدل‌هایی که به‌طور خاص برای شناسایی اشیاء کوچک طراحی شده‌اند (مانند مدل‌هایی با سطوح رزولوشن چندگانه) می‌توانند عملکرد را بهبود دهند.
2. **تنظیم دقیق آستانه IoU:**
   * کاهش آستانه IoU می‌تواند باعث شود مدل پیش‌بینی‌های بیشتری را به عنوان True Positive در نظر بگیرد، که به‌ویژه برای اشیاء کوچک مؤثر است.

### 9-2-2. نمایش نمونه تصویر

با اتمام آموزش و با توجه به ارزیابی مدل با IoU = 0.5 حال می‌توانیم هر عکسی از دیتاست را به مدل داده و پیشبینی مدل و واقعیت را مقایسه کنیم. برای این کار تابعی نوشته شده است که یک عکس از دیتاست به همراه مدل annotation ها و device اجرا که اینجا همان GPU هست را دریافت کرده و روی عکس مشخص می‌کند آنچه در gt هست و آنچه مدل پیشبینی کرده است.



شکل 26 - اجرای تابع نمایش تصاویر و مقایسه پیشبینی مدل و واقعیت برای یک نمونه تصویر



شکل 27- تابع نمایش تصویر و مقایسه بخش اول

A screenshot of a computer program

Description automatically generated

شکل 28 - تابع نمایش تصویر و مقایسه بخش دوم

A road with a bridge and trees

Description automatically generated

شکل 29 - نمونه تصویر مقایسه مدل Faster R-CNN با ground truth

در تصویر فوق همپوشانی 93 درصدی پیشبینی مدل با bounding box موجود در ground truth و همینطور پیشبینی صحیح کلاس را مشاهده می‌کنیم.

## 3-۲.**تنظیم دقیق و ارزیابی مدل تشخیص شی تک مرحله ای**

روش پیاده‌سازی چندین پارامتر و بخش در این مدل مشابه مدل قبل است، بنابراین به جهت پرهیز از تکرار، در بخش‌های مشابه به بخش قبلی ارجاع داده شده است.

### 1-3-2. تعریف شبکه SSD300 و VGG16 به عنوان Backbone

**SSD (Single Shot MultiBox Detector)** یک مدل شناسایی اشیا است که به دلیل سرعت و دقت مناسب برای کاربردهای بلادرنگ بسیار محبوب است. SSD 300 به نسخه‌ای از این مدل اشاره دارد که ورودی تصاویر آن با اندازه‌ی 300 در 300 پیکسل ثابت شده است. این انتخاب باعث کاهش هزینه محاسباتی می‌شود در حالی که دقت کافی برای شناسایی اشیا حفظ می‌شود.

**Backbone: VGG16**

**VGG16** به‌عنوان یک شبکه عصبی از پیش‌آموزش‌دیده عمل می‌کند که نقش استخراج ویژگی‌ها را در SSD برعهده دارد. این شبکه به دلیل معماری عمیق و توانایی در استخراج ویژگی‌های قوی، در بسیاری از مسائل بینایی ماشین استفاده می‌شود. در SSD، لایه‌های fully connected این شبکه حذف می‌شود و لایه‌های اضافی برای شناسایی چندمقیاسی اضافه می‌شوند.

**معماری و روند عملکرد**

1. **پردازش تصویر ورودی**:
   * ورودی مدل یک تصویر 300x300 RGB است.
   * این اندازه ثابت باعث ساده‌سازی فرآیند آموزش و کاهش نیازهای محاسباتی می‌شود.
2. **استخراج ویژگی‌ها**:
   * شبکه VGG16 تصویر ورودی را پردازش کرده و نقشه‌های ویژگی مختلفی را استخراج می‌کند.
   * از برخی از این نقشه‌های ویژگی در لایه‌های میانی برای شناسایی چندمقیاسی استفاده می‌شود.
3. **لایه‌های اضافی**:
   * پس از لایه‌های VGG16، چندین لایه کانولوشن افزوده می‌شود که ابعاد نقشه‌های ویژگی را به تدریج کاهش می‌دهند.
   * این لایه‌ها یک هرم ویژگی ایجاد می‌کنند که مدل را قادر به شناسایی اشیا با اندازه‌های مختلف می‌سازد.
4. **جعبه‌های پیش‌فرض (Default Boxes)**:
   * SSD از جعبه‌های پیش‌فرض با نسبت‌ها و اندازه‌های مختلف در هر سلول نقشه‌های ویژگی استفاده می‌کند.
   * این جعبه‌ها در زمان آموزش با جعبه‌های مرجع (Ground Truth) تطبیق داده می‌شوند.
5. **کلاسه‌بندی و مکان‌یابی**:
   * مدل برای هر جعبه پیش‌فرض دو پیش‌بینی انجام می‌دهد:
     + احتمال تعلق به کلاس‌های مختلف.
     + مختصات جعبه برای تنظیم دقیق موقعیت آن.
   * این پیش‌بینی‌ها با استفاده از هسته‌های کوچک کانولوشن (مثل 3x3) انجام می‌شوند.
6. **تابع هزینه (Loss Function)**:
   * تابع هزینه شامل دو بخش است:
     + **هزینه کلاسه‌بندی**: بر اساس Cross-Entropy.
     + **هزینه مکان‌یابی**: بر اساس Smooth L1 Loss.
7. **حذف حداکثرهای غیرضروری (NMS)**:
   * پس از پیش‌بینی، برای حذف تشخیص‌های اضافی که همپوشانی زیادی دارند، از NMS استفاده می‌شود و جعبه‌ای با بالاترین امتیاز نگه داشته می‌شود.

**مزایای SSD**

* **سرعت بالا**: استفاده از روش تک‌مرحله‌ای باعث حذف شبکه‌های پیشنهاد منطقه و در نتیجه کاهش زمان پردازش می‌شود.
* **تشخیص چندمقیاسی**: استفاده از نقشه‌های ویژگی در چندین مقیاس توانایی مدل در شناسایی اشیا با اندازه‌های مختلف را افزایش می‌دهد.
* **معماری فشرده**: اندازه ورودی کوچک (300x300) نیازهای محاسباتی را کاهش داده و SSD را برای دستگاه‌های کم‌منابع مناسب می‌کند.

**نقش VGG16 در SSD**

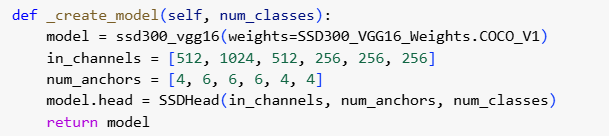
* **ویژگی‌های از پیش‌آموزش‌دیده**: VGG16 که روی دیتاست‌هایی مثل ImageNet آموزش داده شده، ویژگی‌های اولیه‌ی قوی را برای مدل فراهم می‌کند.
* **ویژگی‌های سلسله‌مراتبی**: لایه‌های عمیق VGG16 الگوهای پیچیده را استخراج می‌کنند که برای تشخیص دقیق اشیا حیاتی است.
* **حذف و تطبیق**: با حذف لایه‌های fully connected و افزودن لایه‌های جدید، VGG16 برای شناسایی تراکم بالا (dense prediction) بهینه می‌شود.

**خلاصه روند کار**

1. دریافت یک تصویر ورودی 300 × 300
2. پردازش تصویر توسط VGG16 برای استخراج ویژگی‌ها.
3. اعمال لایه‌های کانولوشن اضافی برای تشخیص چندمقیاسی.
4. تولید پیش‌بینی‌ها برای احتمال کلاسه‌ها و مختصات جعبه.

این معماری در کاربردهایی مثل رانندگی خودکار، نظارت، و استفاده در دستگاه‌های همراه که نیازمند شناسایی اشیا به‌صورت بلادرنگ هستند، استفاده می‌شود.

### 2-3-2. آماده‌سازی اولیه برای تنظیم دقیق



شکل 30- آماده‌سازی مدل برای مدل SSD

مدل SSD300 با معماری VGG16 و وزن‌های از پیش آموزش‌دیده شده روی مجموعه داده COCO بارگذاری شده است. روند کلی مشابه بخش قبلی است است.

هد اصلی مدل با SSDHead جدیدی جایگزین شده است که تعداد کانال‌های ورودی (in\_channels)، تعداد انکرها (num\_anchors) و تعداد کلاس‌های دلخواه را می‌پذیرد.

**دلیل:** این تنظیمات به ما امکان می‌دهد مدل را برای داده‌های جدید (دیتاست سفارشی) بازآموزی کنیم و تنها پارامترهای مربوط به دسته‌بندی‌ها و تشخیص روی داده‌های ما تغییر کنند.

**1. in\_channels:**

* این لیست نشان‌دهنده تعداد **کانال‌های ویژگی** است که از لایه‌های مختلف شبکه استخراج می‌شود.

**توضیح:**

* در SSD، **لایه‌های خاصی از شبکه عصبی (مانند VGG16)** برای استخراج ویژگی‌های مکانی و مفهومی اشیاء استفاده می‌شوند.
* این لایه‌ها معمولاً لایه‌های آخر شبکه هستند که ویژگی‌های **انتزاعی‌تر و بزرگ‌تر** از تصویر ارائه می‌دهند.
* هر لایه خروجی (feature map) تعدادی کانال دارد که این کانال‌ها نشان‌دهنده عمق اطلاعات استخراج‌شده از تصویر هستند.
* به عنوان مثال:
  + **512 کانال:** به این معنی است که در لایه‌ای خاص، 512 ویژگی مختلف (فیلتر یا کرنل) از تصویر استخراج شده است.
  + این اعداد (512، 1024، ...) مستقیماً از معماری مدل اصلی (VGG16) به دست آمده‌اند.

**چرا این اعداد انتخاب شده‌اند؟**

* این اعداد از ساختار معماری **VGG16** می‌آیند:
  + 512 و 1024 کانال‌ها از لایه‌های آخر VGG16 هستند.
  + لایه‌های اضافی با ابعاد کوچک‌تر (256 کانال) به مدل اضافه می‌شوند تا از مقیاس‌های مختلف تصویر استفاده شود. این لایه‌ها در اصل بخشی از معماری SSD هستند.

**2. num\_anchors:**

* این لیست تعداد **انکر باکس‌ها** را در هر لایه ویژگی نشان می‌دهد.

**توضیح:**

* **انکر باکس‌ها (Anchor Boxes):**
  + در SSD، برای هر سلول در feature map، چندین باکس پیش‌فرض( default boxes یا anchor boxes ) با نسبت ابعاد (aspect ratios) مختلف تعریف می‌شود.
  + این باکس‌ها کمک می‌کنند که مدل اشیاء با اندازه‌ها و نسبت‌های مختلف را تشخیص دهد.
* **چرا تعداد انکرها متفاوت است؟**
  + هر لایه ویژگی (feature map) اندازه متفاوتی دارد و برای پوشش ابعاد مختلف اشیاء، تعداد متفاوتی انکر تعریف شده است.
  + به عنوان مثال:
    - لایه‌ای که ویژگی‌های کوچک‌تر را استخراج می‌کند، معمولاً نیاز به تعداد بیشتری انکر دارد (مثل 6 انکر برای پوشش ابعاد مختلف).
    - در لایه‌هایی که مقیاس بزرگ‌تری دارند، تعداد انکرها کمتر است، زیرا اشیاء بزرگ‌تر و ساده‌تری را پوشش می‌دهند.

**مقادیر:**

* **4 انکر:** برای لایه‌هایی که اشیاء کوچک‌تر را پوشش می‌دهند.
* **6 انکر:** برای لایه‌هایی که اشیاء بزرگ‌تر یا پیچیده‌تر را پوشش می‌دهند.

**چگونه این اعداد تعیین می‌شوند؟**

* این اعداد از طراحی معماری SSD و استاندارد COCO یا Pascal VOC برای تشخیص اشیاء می‌آیند. در SSD300، مقادیر استاندارد 4 و 6 برای انکرها استفاده می‌شوند.
* هر انکر یک جعبه با نسبت ابعاد (aspect ratio) متفاوت دارد (مثل 1:1، 1:2، 2:1 و غیره).

**جمع‌بندی:**

1. **in\_channels:**
   * این اعداد از معماری **VGG16** می‌آیند و تعداد کانال‌های لایه‌های مختلف شبکه را نشان می‌دهند که برای استخراج ویژگی استفاده می‌شوند.
2. **num\_anchors:**
   * تعداد انکر باکس‌ها در هر لایه feature map است. این تعداد برای پوشش نسبت ابعاد مختلف اشیاء در مقیاس‌های متفاوت انتخاب شده و از طراحی استاندارد SSD300 تبعیت می‌کند.

**چرا این دو بخش مهم‌اند؟**

* این مقادیر به طور مستقیم به **اندازه و مقیاس اشیاء** در تصاویر ورودی مربوط می‌شوند.
* اگر این مقادیر درست تنظیم شوند، مدل می‌تواند **اشیاء با اندازه‌های مختلف** را بهتر شناسایی کند.

### 3-3-2. عملیات آماده‌سازی و نرمال کردن

نکته حائز اهمیت و متفاوت در این مدل سایز تصاویر است. تصاویر در این مدل نهایتا همگی 300 در 300 خواهند شد. می‌توان پیش از شروع train و به صورت خارجی عکس‌ها را Resize کرد و یا تصاویر را با ابعاد اولیه و واقعی به مدل بدهیم و عکس‌ها در داخل مدل و به صورت داخلی و اتوماتیک ریسایز شوند. که ما گزینه دوم را انتخاب کردیم چرا که ریسایز کردن عکس‌ها قبل از آموزش باعث می‌شود در زمان ارزیابی با سایز نیاز باشد مجدد آنها را به سایزهای واقعی برگردانیم تا امکان ارزیابی به صورت صحیح مهیا باشد. همچنین با انجام چند بار تست راه دوم دقت بالاتری را نتیجه داد.

ابتدا مدل بدون داده افزایی خاصی مدل شد که نتیجه چندان قابل قبول نبود به همین دلیل اقدام به استفاده از augmentation های متفاوت کردیم. به عنوان مثال تغییر در نور و saturation انجام شد اما نتیجه به اندازه کافی خوب نبود. در نهایت با تکرار چندید حالت مختلف از augmentation (به صورت داینامیک) به این نتیجه رسیدیم که با horizontal flip به بهترین نتیجه می‌رسیم. در بخش ارزیابی نتیجه این اعمال داده افزایی نمایش داده خواهد شد. در بخش horizontal flip به جا به جایی bounding box ها هم توجه شده است به این معنا که عکسی که به صورت افقی flip می‌شود طبیعتا محل bounding box هم برای آن باید تغییر کند و متناظر با عکس در حالت جدید تنظیم شود.



شکل 31 - نرمال‌سازی و آماده سازی تصاویر برای آموزش

در ادامه کلاسی به نام GTSDBDataset داریم که بسیار مشابه بخش قبلی است و در ساختار هیچ تفاوتی وجود ندارد و نهایتا دیتاست مورد نیاز را آماده می‌کند. استفاده از DataLoader و باقی موارد این قسمت هم دقیقا مشابه بخش قبلی است.

### 4-3-2. معیارهای IoU و mAP

تعریف و دلیل استفاده از این معیار در بخش قبلی آورده شده است. نحوه پیاده‌سازی همچنین دقیقا مشابه بخش قبل است.

### 5-3-2. بهینه‌ساز و تابع هزینه

**تابع هزینه**

برای تابع هزینه یا loss از آنچه در درون خود SSD300 بوده استفاده کرده ایم که به شکل زیر پیاده سازی شده است:

خطا شامل دو بخش است:

* **خطای کلاسه‌بندی**: خطای Cross-Entropy برای پیش‌بینی چندکلاسه.
* **خطای مکان‌یابی**: خطای Smooth L1 برای تنظیم موقعیت جعبه.

دلیل استفاده از این توابع برای تابع هزینه دقیقا مشابه بخش قبلی است. به همین جهت برای مطالعه علت استفاده از این توابع به بخش قبلی با همین عنوان مراجعه بفرمایید.

**بهینه‌ساز**

برای بهینه‌ساز از **Adam** استفاده شده است. همچنین از **lr\_scheduler** نیز بهره گرفته شده تا امکان کاهش و تنظیم نرخ یادگیری بهینه‌تر فراهم شود و عملکرد مدل در طول آموزش بهبود یابد.

**ویژگی‌های Adam:**

1. **ترکیب نرخ یادگیری تطبیقی (Adaptive Learning Rate):** 
   * Adam از نرخ یادگیری تطبیقی استفاده می‌کند، به این معنا که برای هر پارامتر نرخ یادگیری جداگانه و بهینه در نظر گرفته می‌شود.
   * این ویژگی باعث می‌شود مدل سریع‌تر و با دقت بالاتر به حداقل سراسری (Global Minimum) برسد.
2. **نرخ یادگیری پایین (0.0001):** 
   * نرخ یادگیری اولیه برابر با 0.0001 تنظیم شده است تا مدل پیش‌آموزش‌دیده به طور دقیق تنظیم (Fine-Tuned) شود.
   * این مقدار به مدل کمک می‌کند که پارامترهای خود را به تدریج و بدون تغییرات ناگهانی بهبود دهد.
3. **وزن‌زدایی (Weight Decay) :** 
   * مقدار Weight Decay برابر با 0.0005 در نظر گرفته شده است.
   * این ویژگی باعث جلوگیری از بیش‌برازش (Overfitting) می‌شود، به‌ویژه در مدل‌هایی مانند SSD که تعداد زیادی پارامتر دارند.

**دلایل انتخاب Adam به عنوان بهینه‌ساز:**

**1. Adaptive Learning Rate**

* **چرا انتخاب شده؟**
  + Adam نرخ یادگیری تطبیقی دارد، که به مدل کمک می‌کند برای هر پارامتر نرخ یادگیری بهینه تعیین کند.
  + این ویژگی به‌خصوص در مدل‌های پیچیده مثل SSD که دارای تعداد زیادی پارامتر هستند، بسیار مفید است.
* **مزیت:** بهبود سرعت و دقت همگرایی (Convergence).

**2. وزن‌زدایی (Weight Decay)**

* **چرا مهم است؟**
  + Weight Decay به‌عنوان یک روش مؤثر برای جلوگیری از بیش‌برازش عمل می‌کند.
  + در Adam، این ویژگی به‌طور مستقیم بر روی به‌روزرسانی پارامترها تأثیر می‌گذارد و شبکه را بهینه‌تر تنظیم می‌کند.
* **مزیت:** جلوگیری از بیش‌برازش و بهبود تعمیم‌دهی (Generalization) مدل.

**3.ترکیب مومنتوم (Momentum)**

* **چرا مفید است؟**
  + مومنتوم به مدل کمک می‌کند که از کمینه‌های محلی (Local Minima) عبور کرده و به کمینه سراسری برسد.
  + این ویژگی در مسائل پیچیده مانند شناسایی اشیاء، که ممکن است تابع هدف دارای چندین کمینه محلی باشد، بسیار ارزشمند است.
* **مزیت:** افزایش پایداری در آموزش و همگرایی بهتر.

**4. پایداری در آموزش**

* **چرا انتخاب شده؟**
  + Adam در مسائل با داده‌های پیچیده و حجیم (مثل مجموعه داده‌های COCO و GTSDB رفتار پایداری دارد.
  + این پایداری ناشی از ترکیب نرخ یادگیری تطبیقی و مومنتوم مناسب است.
* **مزیت:** عملکرد بهتر روی داده‌های متنوع و پیچیده.

**چگونگی ترکیب با lr\_scheduler**

* از **lr\_scheduler.MultiStepLR** برای تنظیم نرخ یادگیری استفاده شده است:

lr\_scheduler = torch.optim.lr\_scheduler.MultiStepLR(optimizer, milestones=[4, 8], gamma=0.1)

* **هدف:**
  + کاهش نرخ یادگیری در دوره‌های مشخص (Milestones) برای کمک به همگرایی بهتر.
  + این روش به مدل اجازه می‌دهد در مراحل اولیه سریع‌تر یاد بگیرد و در مراحل پایانی با دقت بیشتری تنظیم شود.

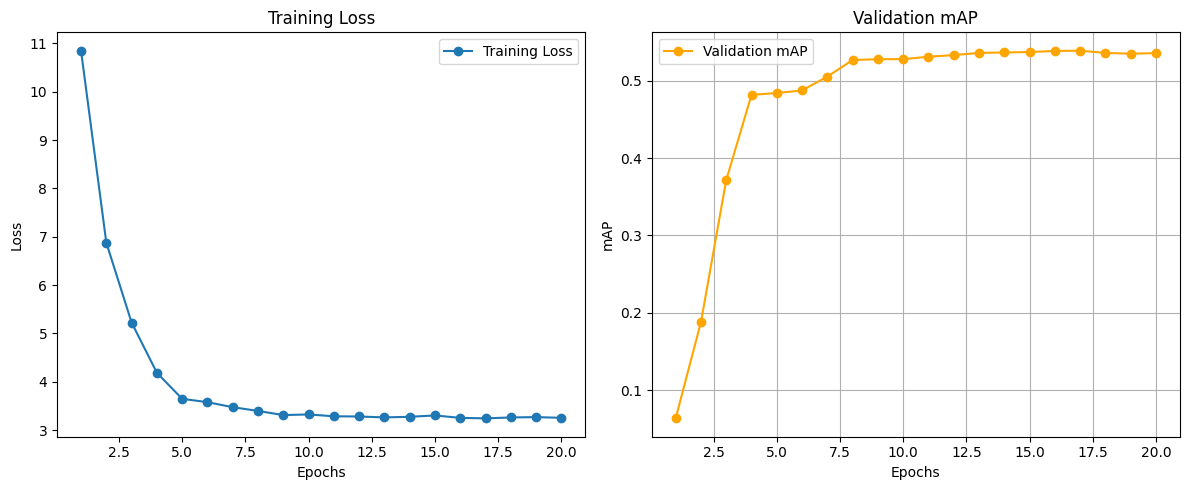
**در مرحله آموزش فرض‌هایی صورت گرفته است که به آنها اشاره می‌شود:**

1. **تعداد epoch:** این مقدار در مقاله یا صورت سوال اشاره نشده است و ما با توجه به زمانبر بودن اجرا و دست یافتن به دقت قابل قبول، تعداد 20 ایپاک را در نظر گرفتیم.
2. **Batch size:** این مقدار هم مشخصا تعریف نشده است. برای این سوال عدد 8 را برای batch size در نظر گرفته‌ایم.

اکنون مدل آماده آموزش است و با استفاده از تابع زیر که در کلاس TrafficSignDetectorSSD تعریف شده این کار را انجام می‎‌دهیم. برای ارزیابی و چگونگی انجام آن در ادامه توضیح داده شده است.

### 6-3-2. ارزیابی مدل

تابع ارزیابی در این بخش هم دقیقا مشابه بخش قبلی نوشته شده است. از نظر عملکرد و ساختار هیچ تفاوتی با بخش قبلی وجود ندارد اما نتایج به صورت زیر بدست آمده است.



شکل 32- ارزیابی عملکرد مدل SSD300 در حین آموزش با IoU=0.5

روند نزولی loss و صعودی mAP نشان از عملکرد صحیح و مناسب مدل است چرا که می‌دانیم مقدار loss در هر مرحله و با عملیات ریاضیتی داخل optimizer باید کاهش یابد و از طرفی مدل در هر گام باید بهتر از گام قبل عمل کند.



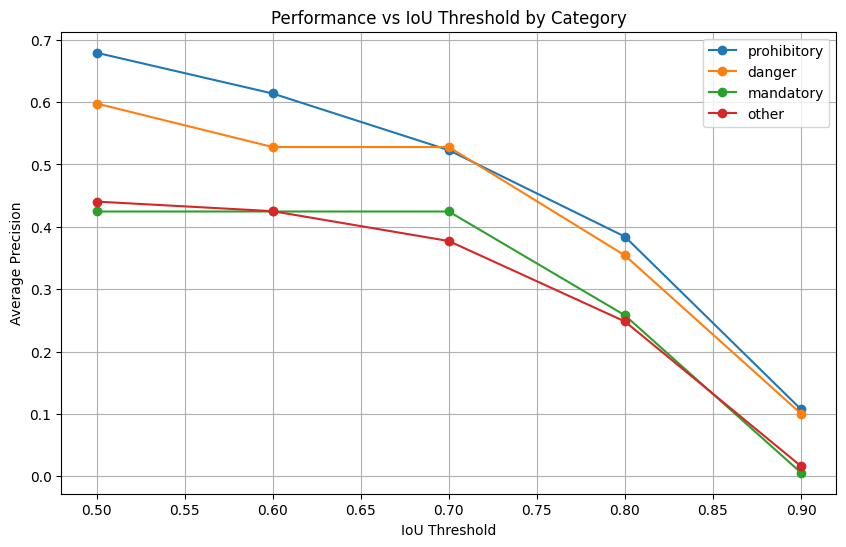
شکل 33- mAP هر کلاس

جدول 4- جدول mAP با معیار IoU = 0.5 برای هر کلاس و به صورت میانگین

|  |  |
| --- | --- |
|  | mAP |
| **Prohibitory**  **Danger** | 67.92  59.78 |
| **Mandatory** | 42.46 |
| **Other** | 44.05 |
| **میانگین** | **53.55** |

### 7-3-2. ترسیم نمودار AP به ازای مقادیر متفاوت IoU

تابع و نحوه پیاده‌سازی این بخش هم دقیقا مشابه بخش قبلی است. اما نمودار و تحلیل آن به شرح زیر است.



شکل 34- نمودار مقادیر مختلف mAP برای هر کلاس با توجه به threshold انتخابی

**1. مشاهده کلی:**

1. **عملکرد دسته‌ها:**
   * دسته **prohibitory** عملکرد بالاتری نسبت به بقیه دسته‌ها دارد. AP برای این دسته در تمام آستانه‌های IoU بالاتر از دیگر دسته‌هاست.
   * دسته **danger** نیز عملکرد مناسبی دارد اما نسبت به دسته prohibitory کمتر است.
   * دسته‌های **mandatory** و **other** عملکرد ضعیف‌تری نسبت به دو دسته بالا دارند و در آستانه‌های IoU بالا کاهش دقت قابل توجهی دارند.
2. **روند کاهش دقت:**
   * با افزایش مقدار IoU از 0.5 به 0.9، میانگین دقت در تمام دسته‌ها کاهش می‌یابد. این کاهش طبیعی است زیرا آستانه IoU بالاتر به معنای سختگیری بیشتر در مطابقت پیش‌بینی مدل با جعبه واقعی (Ground Truth) است.

**2.تحلیل دسته‌ها:**

**الفProhibitory (:**

* **مشاهدات:**
  + این دسته در تمام آستانه‌های IoU بالاترین میانگین دقت را دارد و از حدود 0.7 (IoU = 0.5) شروع شده و تا حدود 0.3 (IoU = 0.9) کاهش می‌یابد.
* **دلیل عملکرد بهتر:**
  + علائم prohibitory معمولاً دارای طراحی ساده‌تر، شکل‌های واضح‌تر (دایره‌ای با رنگ قرمز) و تمایز بیشتری نسبت به پس‌زمینه هستند، که تشخیص آن‌ها را برای مدل آسان‌تر می‌کند.

**بDanger (:**

* **مشاهدات:**
  + این دسته عملکردی نسبتاً خوب دارد، اما کمی پایین‌تر از دسته prohibitory است. AP از حدود 0.6 (IoU = 0.5) شروع شده و به 0.2 (IoU = 0.9) کاهش می‌یابد.
* **دلیل عملکرد:**
  + علائم danger معمولاً مثلثی شکل هستند، اما ممکن است به دلیل پیچیدگی بیشتر طراحی (متن یا تصاویر داخلی)، تشخیص آن‌ها سخت‌تر باشد.

**ج Mandatory (:**

* **مشاهدات:**
  + عملکرد این دسته ضعیف‌تر از دو دسته بالا است. AP از حدود 0.45 (IoU = 0.5) شروع شده و به کمتر از 0.15 (IoU = 0.9) کاهش می‌یابد.
* **دلیل عملکرد ضعیف‌تر:**
  + علائم mandatory ممکن است طراحی پیچیده‌تری داشته باشند یا تنوع بیشتری در اندازه و شکل داشته باشند که تشخیص آن‌ها را دشوارتر می‌کند.

**دOther (:**

* **مشاهدات:**
  + این دسته پایین‌ترین عملکرد را دارد. AP از حدود 0.4 (IoU = 0.5) شروع شده و به نزدیک صفر (IoU = 0.9) کاهش می‌یابد.
* **دلیل عملکرد ضعیف‌تر:**
  + دسته other احتمالاً شامل علائم متفرقه‌ای است که ویژگی‌های مشخصی ندارند یا شباهت زیادی به پس‌زمینه دارند. این می‌تواند منجر به تشخیص نادرست یا از دست رفتن برخی از علائم شود.

**3.تحلیل کاهش دقت با IoU:**

* **کاهش طبیعی:**
  + همان‌طور که IoU افزایش می‌یابد، مدل باید پیش‌بینی‌های دقیق‌تری انجام دهد. بنابراین، میانگین دقت در تمام دسته‌ها کاهش می‌یابد.
* **اختلاف عملکرد:**
  + در IoU بالا (مثل 0.8 و 0.9)، اختلاف میان دسته‌ها بیشتر به چشم می‌آید. دسته‌های prohibitory و danger همچنان بهتر عمل می‌کنند، اما mandatory و other دقت خود را به شدت از دست می‌دهند.

**جمع‌بندی:**

این نمودار نشان می‌دهد که مدل در شناسایی علائم prohibitory و danger عملکرد خوبی دارد، اما برای دسته‌های mandatory و other نیاز به بهبود وجود دارد. افزایش داده‌های آموزشی، تغییر انکرها و تنظیم دقیق‌تر مدل می‌تواند به بهبود این دسته‌ها کمک کند.

جدول 5 - ارزیابی مدل SSD300 با threshold های متفاوت

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | mAP | | | | |
|  | IoU = 0.9 | IoU = 0.8 | IoU = 0.7 | IoU = 0.6 | IoU = 0.5 |
| **Prohibitory** | 10.73 | 38.42 | 52.29 | 61.38 | 67.92 |
| **Danger** | 10.03 | 35.41 | 52.80 | 52.80 | 59.78 |
| **Mandatory** | 5.66 | 25.81 | 42.46 | 42.46 | 42.46 |
| **Other** | 1.62 | 24.82 | 42.51 | 42.51 | 44.05 |
| **میانگین** | **5.98** | **31.12** | **46.32** | **49.77** | **53.55** |

دقت مدل کمتر از میزان قابل پیشبینی ما بوده است که در بخش ارزیابی و مقایسه نتایج دو مدل به شکل مفصلی در رابطه با علت کم بودن دقت نسبت به مدل دیگر و بالاتر نیامدن آن صحبت شده است.

### 8-3-2. ارزیابی مدل بر اساس اندازه

تابع نوشته شده و نحوه اجرا دقیقا مشابه بخش قبلی است اما نمودار این بخش و تحیل آن به فرم زیر است.

A graph of a performance by object size

Description automatically generated

شکل 35- ارزیابی مدل SSD300 بر اساس سایز

جدول 6- ارزیابی مدل SSD300 بر اساس سایز

|  |  |  |  |
| --- | --- | --- | --- |
|  | mAP for IoU = 0.5 | | |
|  | Large | Medium | Small |
| mAP for IoU = 0.5 | **95.87** | **62.02** | **8.64** |

این نمودار نشان‌دهنده عملکرد مدل در تشخیص اشیاء با اندازه‌های مختلف است. محور افقی سه دسته اندازه اشیاء small، medium، large را نشان می‌دهد و محور عمودی مقدار mAP (میانگین دقت) را برای هر دسته اندازه مشخص می‌کند. تحلیل دقیق نمودار به شرح زیر است:

**1.مشاهده کلی:**

1. **اشیاء کوچک (small):**
   * این دسته پایین‌ترین عملکرد را دارد، با مقدار mAP کمتر از **0.2**.
   * این نشان می‌دهد که مدل در شناسایی اشیاء کوچک به طور قابل توجهی ضعیف‌تر عمل می‌کند.
2. **اشیاء متوسط (medium):**
   * این دسته عملکرد بهتری نسبت به اشیاء کوچک دارد و mAP آن در حدود **0.6** است.
   * مدل در این دسته به شکل قابل قبولی عمل می‌کند، اما همچنان بهبودهایی ممکن است.
3. **اشیاء بزرگ (large):**
   * این دسته بهترین عملکرد را با mAP نزدیک به **1.0** دارد.
   * مدل در شناسایی اشیاء بزرگ عملکرد بسیار خوبی داشته و تقریباً به دقت کامل رسیده است.

**2.تحلیل دسته‌ها:**

**الف. اشیاء کوچک (small):**

* **مشاهدات:**
  + مدل در شناسایی اشیاء کوچک ضعیف عمل می‌کند.
* **دلیل عملکرد ضعیف:**
  + اشیاء کوچک معمولاً وضوح کمتری دارند و ممکن است در تصاویر نویزی یا دارای پس‌زمینه شلوغ به خوبی قابل تشخیص نباشند.
  + ابعاد کوچک این اشیاء ممکن است باعث شود انکر باکس‌های پیش‌فرض مدل با این اشیاء هم‌خوانی نداشته باشند.

**ب. اشیاء متوسط (medium):**

* **مشاهدات:**
  + عملکرد مدل در شناسایی اشیاء متوسط قابل قبول است، اما هنوز جای بهبود دارد.
* **دلیل عملکرد بهتر:**
  + اشیاء متوسط به اندازه کافی بزرگ هستند که جزئیات آن‌ها در تصاویر به وضوح قابل مشاهده باشد.
  + انکر باکس‌های پیش‌فرض مدل به طور مؤثری می‌توانند این اشیاء را پوشش دهند.

**ج. اشیاء بزرگ (large):**

* **مشاهدات:**
  + مدل در شناسایی اشیاء بزرگ عملکرد تقریباً بی‌نقص دارد.
* **دلیل عملکرد عالی:**
  + اشیاء بزرگ در تصاویر به وضوح قابل مشاهده هستند و انکر باکس‌ها به راحتی می‌توانند با این اشیاء تطابق داشته باشند.
  + این اشیاء معمولاً نویز کمتری دارند و شباهت کمتری به پس‌زمینه دارند.

**3.جمع‌بندی:**

* **اشیاء بزرگ:** عملکرد عالی با mAP نزدیک به 1.0.
* **اشیاء متوسط:** عملکرد خوب اما با امکان بهبود.
* **اشیاء کوچک:** عملکرد ضعیف، نیازمند تمرکز بیشتر مدل بر این دسته است.

### 9-3-2. نمایش نمونه تصویر

باز هم تابع نوشته شده و نحوه استفاده دقیقا مشابه بخش قبل است. اما تصویر نمونه برای این قسمت به فرم زیر است.



شکل 36- نمونه تصویر مقایسه مدل SSD300 با ground truth

در تصویر فوق همپوشانی 75 درصدی پیشبینی مدل با bounding box موجود در ground truth و همینطور پیشبینی صحیح کلاس را مشاهده می‌کنیم.

## 4-۲. **ارزیابی نتایج و مقایسه مدل‌ها**

### 2-4-1. مقایسه مدل‌های Faster R-CNN و SSD بر اساس آستانه‌های IoUو برای کلاس‌های متخلف

**مقایسه کلی:**

1. **Faster R-CNN:** 
   * عملکرد بهتری در تمام آستانه‌های IoU، به خصوص در مقادیر بالاتر (0.8 و 0.9) دارد.
   * مقدار mAP بالایی را برای اکثر دسته‌ها در آستانه‌های پایین‌تر IoU حفظ می‌کند.
2. **SSD:**
   * به طور کلی عملکرد ضعیف‌تری دارد و با افزایش آستانه IoU کاهش چشم‌گیری در عملکرد آن دیده می‌شود.
   * به ویژه در دسته‌های "mandatory" و "other" ضعیف‌تر عمل می‌کند.

**عملکرد به تفکیک دسته‌ها:**

**Prohibitory:**

* **Faster R-CNN:** 
  + mAP بالایی در تمام آستانه‌ها نشان می‌دهد؛ از مقدار 0.972 در IoU=0.5 شروع شده و به 0.305 در IoU=0.9 کاهش می‌یابد.
  + پایداری بهتری نسبت به SSD دارد.
* **SSD:** 
  + mAP در IoU=0.5 برابر با 0.679 است، اما در IoU=0.9 به 0.107 کاهش می‌یابد.
* **برنده:**  Faster R-CNN
  + این مدل دقت و پایداری بیشتری برای این دسته دارد.

**Danger :**

* **Faster R-CNN:**
  + عملکرد خوبی دارد؛ از 0.951 در IoU=0.5 شروع شده و به 0.266 در IoU=0.9 کاهش می‌یابد.
  + در تمام آستانه‌ها mAP بالاتری نسبت به SSD دارد.
* **SSD:** 
  + mAP از 0.597 در IoU=0.5 شروع شده و در IoU=0.9 به 0.100 می‌رسد.
* **برنده:** Faster R-CNN
  + به دلیل بازدهی بهتر در تشخیص دقیق‌تر این دسته.

**Mandatory :**

* **Faster R-CNN:** 
  + از 0.884 در IoU=0.5 شروع شده و به 0.247 در IoU=0.9 کاهش می‌یابد.
  + عملکرد نسبتاً پایدار در این دسته دارد.
* **SSD**
  + عملکرد ضعیفی دارد؛ از 0.424 در IoU=0.5 شروع شده و به 0.005 در IoU=0.9 کاهش می‌یابد.
* **برنده:** Faster R-CNN
  + SSD به وضوح در تشخیص اشیاء کوچک‌تر یا پیچیده‌تر در این دسته ضعف دارد.

**Other :**

* **Faster R-CNN:**
  + mAP از 0.923 در IoU=0.5 شروع شده و به 0.141 در IoU=0.9 کاهش می‌یابد.
  + در تمام آستانه‌ها بهتر از SSD عمل می‌کند.
* **SSD:** 
  + mAP از 0.440 در IoU=0.5 شروع شده و در IoU=0.9 به 0.016 کاهش می‌یابد.
* **برنده:** Faster R-CNN
  + عملکرد ضعیف SSD نشان‌دهنده دشواری آن در شناسایی اشیاء متنوع و نامعمول است.

**مقایسه mAP کلی:**

**Faster R-CNN:**

* مقدار mAP در تمام آستانه‌ها بالاتر است:
  + **IoU=0.5:** 0.932
  + **IoU=0.9:** 0.240

**SSD:**

* mAP کلی پایین‌تر است و کاهش بیشتری با افزایش IoU دارد:
  + **IoU=0.5:** 0.535
  + **IoU=0.9:** 0.057
* **برنده:**  Faster R-CNN
  + این مدل در مجموع دقت بالاتری دارد و پایداری بیشتری در آستانه‌های IoU مختلف نشان می‌دهد.

**دلایل تفاوت عملکرد:**

**چرا Faster R-CNN بهتر عمل می‌کند؟**

1. **شبکه پیشنهادی منطقه‌ای (RPN):**
   * Faster R-CNN از یک شبکه پیشنهادی برای تولید جعبه‌های منطقه‌ای (Region Proposals) استفاده می‌کند که باعث شناسایی دقیق‌تر اشیاء، به ویژه در آستانه‌های بالای IoU می‌شود.
   * این ویژگی به بهبود تشخیص دسته‌هایی مانند "mandatory" و "other" کمک می‌کند.
2. **استخراج ویژگی‌های قوی‌تر:**

* Faster R-CNN از یک شبکه عمیق‌تر و پیشرفته‌تر (مانند ResNet) برای استخراج ویژگی استفاده می‌کند، که توانایی آن را در شناسایی اشیاء پیچیده و جزئیات ظریف افزایش می‌دهد.

1. **تنظیم دقیق جعبه‌ها:**

* این مدل دقت بالایی در پیش‌بینی مختصات جعبه‌های محدودکننده (Bounding Box) دارد که به حفظ دقت در آستانه‌های IoU بالاتر کمک می‌کند.

1. **مدیریت بهتر مقیاس‌ها:**

* معماری Faster R-CNN به طور طبیعی برای مدیریت اشیاء با اندازه‌ها و مقیاس‌های مختلف مناسب‌تر است.

1. **سازگاری بیشتر با داده‌های کم:**

* Faster R-CNN به دلیل معماری پیشرفته‌تر و استفاده از ویژگی‌های عمیق‌تر، عملکرد پایدارتری در مجموعه داده‌های کوچک (مانند 900 تصویر شما) دارد.
* شبکه پیشنهادی (RPN) این مدل نیاز به داده‌های کمتر برای یادگیری توزیع جعبه‌های پیشنهادی دارد، که آن را برای داده‌های کم مناسب‌تر می‌کند.

**چرا SSD عملکرد ضعیف‌تری دارد؟**

1. **باکس‌های انکر ثابت:**
   * SSD به شدت به باکس‌های انکر از پیش تعریف‌شده متکی است، که ممکن است با اندازه‌ها و نسبت‌های متفاوت اشیاء به خوبی تطبیق نداشته باشد، به ویژه برای اشیاء کوچک‌تر.
2. **ویژگی‌های کم‌عمق‌تر:**
   * SSD از نقشه‌های ویژگی کم‌عمق‌تر برای پیش‌بینی استفاده می‌کند، که توانایی آن را در استخراج ویژگی‌های پیچیده محدود می‌کند.
3. **عدم وجود RPN:**
   * نبود شبکه پیشنهادی منطقه‌ای باعث می‌شود SSD پیش‌بینی‌های مستقیم انجام دهد، که دقت را کاهش داده و منجر به افزایش مثبت‌های کاذب می‌شود.
4. **حساسیت به اشیاء کوچک:**
   * SSD در شناسایی اشیاء کوچک یا دارای ویژگی‌های ظریف، مانند دسته "mandatory"، به وضوح مشکل دارد.
5. **نیاز به داده‌های بیشتر:**
   * SSD برای یادگیری توزیع جعبه‌ها و تنظیم انکرها نیاز به تعداد زیادی داده آموزشی دارد. با توجه به اینکه تعداد داده‌های شما تنها **900 تصویر** است، این مدل به دلیل معماری سبک‌تر و وابستگی به داده‌های بزرگ، عملکرد ضعیف‌تری نشان می‌دهد.

پس به صورت کلی می‌توان گفت:

* **Faster R-CNN:** به دلیل استفاده از RPN و معماری عمیق، توانایی بهتری در تعمیم‌دهی با داده‌های کم دارد. این موضوع باعث می‌شود حتی با تعداد محدود تصاویر (900 تصویر)، مدل عملکرد مناسبی داشته باشد.
* **SSD:** به داده‌های بیشتر برای یادگیری بهتر نیاز دارد. تعداد محدود تصاویر باعث می‌شود این مدل نتواند انکرها را به خوبی تطبیق داده و اشیاء را به دقت تشخیص دهد.

**نتیجه‌گیری نهایی:**

* **Faster R-CNN** به دلیل سازگاری بیشتر با مجموعه داده‌های کوچک (900 تصویر) و معماری پیشرفته‌تر، انتخاب بهتری برای شناسایی دقیق علائم راهنمایی و رانندگی است.
* **SSD** با وجود سبک‌تر بودن، برای مجموعه داده‌های کوچک مناسب نیست و دقت آن به شدت کاهش می‌یابد.

### 2-4-2. مقایسه مدل‌های Faster R-CNN و SSD300 بر اساس اندازه اشیاء

**1. مقایسه کلی:**

1. **Faster R-CNN:**
   * **اشیاء کوچک:**  0.846
   * **اشیاء متوسط:**  0.902
   * **اشیاء بزرگ:**  0.958
   * عملکردی یکنواخت و قوی در تمام اندازه‌ها دارد، با کاهش جزئی در اشیاء کوچک.
2. **SSD300:**
   * **اشیاء کوچک:**  0.086
   * **اشیاء متوسط:**  0.620
   * **اشیاء بزرگ:**  0.958
   * عملکرد بسیار ضعیف در اشیاء کوچک، متوسط برای اشیاء متوسط، و برابر با Faster R-CNN برای اشیاء بزرگ.

**2.عملکرد بر اساس اندازه اشیاء:**

**اشیاء کوچک:**

* **Faster R-CNN:**
  + mAP: 0.846
  + عملکرد بسیار عالی در شناسایی اشیاء کوچک.
* **SSD300:**
  + mAP: 0.086
  + عملکرد بسیار ضعیف که نشان‌دهنده مشکل مدل در شناسایی اشیاء کوچک است.
* **برنده:** Faster R-CNN

**اشیاء متوسط:**

* **Faster R-CNN:** 
  + mAP: 0.902
  + عملکرد بسیار خوب با کاهش جزئی نسبت به اشیاء بزرگ.
* **SSD300:**
  + mAP: 0.620
  + عملکرد متوسط اما به طور قابل توجهی پایین‌تر از Faster R-CNN.
* **برنده:** Faster R-CNN

**اشیاء بزرگ:**

* **Faster R-CNN:**
  + mAP: 0.958
  + عملکرد عالی و مشابه SSD300.
* **SSD300:** 
  + mAP: 0.958
  + عملکرد برابر با Faster R-CNN.
* **برنده:** مساوی

**3. جمع‌بندی کلی:**

* **Faster R-CNN** : عملکرد کلی بهتری در تمام اندازه‌ها دارد و در شناسایی اشیاء کوچک و متوسط بسیار بهتر عمل می‌کند.
* **SSD300** : تنها در شناسایی اشیاء بزرگ عملکرد قابل قبولی دارد، اما در اشیاء کوچک و متوسط بسیار ضعیف‌تر است.

**دلایل تفاوت عملکرد:**

**چرا Faster R-CNN بهتر عمل می‌کند؟**

1. **شبکه پیشنهادی منطقه‌ای (RPN):** 
   * استفاده از RPN باعث شناسایی دقیق‌تر اشیاء به خصوص برای اندازه‌های کوچک و متوسط می‌شود.
2. **استخراج ویژگی‌های عمیق‌تر:**
   * Faster R-CNN از شبکه‌های عمیق‌تر برای استخراج ویژگی استفاده می‌کند که برای شناسایی جزئیات اشیاء کوچک مناسب‌تر است.
3. **تنظیم دقیق جعبه‌ها:**
   * روش دو مرحله‌ای این مدل باعث تنظیم بهتر جعبه‌ها می‌شود.
4. **سازگاری بیشتر با داده‌های کم:**
   * با توجه به تعداد کم تصاویر (900 عکس)، این مدل بهتر از SSD می‌تواند تعمیم دهد و یاد بگیرد.

**چرا SSD300 عملکرد ضعیف‌تری دارد؟**

1. **باکس‌های انکر ثابت:**
   * SSD نمی‌تواند اشیاء کوچک یا متوسط را به خوبی با انکرهای پیش‌فرض شناسایی کند.
2. **ویژگی‌های کم‌عمق‌تر:**
   * ویژگی‌های کم‌عمق باعث می‌شود مدل در استخراج جزئیات اشیاء کوچک ضعیف عمل کند.
3. **عدم وجود RPN:**
   * نبود شبکه پیشنهادی باعث کاهش دقت و افزایش مثبت‌های کاذب می‌شود.
4. **نیاز به داده‌های بیشتر:**
   * SSD برای یادگیری بهتر به تعداد بیشتری داده نیاز دارد و تعداد محدود (900 تصویر) برای آن کافی نیست.

**نتیجه‌گیری نهایی:**

* **Faster R-CNN** به دلیل معماری پیشرفته‌تر، برای این مجموعه داده کوچک (900 عکس) و شناسایی اشیاء کوچک و متوسط، انتخاب مناسب‌تری است.
* **SSD300** با وجود سبک‌تر بودن، برای مجموعه داده‌های کوچک مناسب نیست و در شناسایی دقیق اشیاء کوچک به شدت ضعف دارد.

### 2-4-3. عوامل مؤثر بر عملکرد مدل‌ها

عملکرد مدل‌ها می‌تواند تحت تأثیر عوامل مختلفی قرار گیرد. در اینجا مهم‌ترین عوامل را بررسی می‌کنیم:

**1.کیفیت و تعداد داده‌ها**

* **تأثیر:**
  + تعداد کم داده‌ها (مانند 900 تصویر در این مورد) می‌تواند منجر به یادگیری ناکافی مدل شود، به خصوص برای مدل‌هایی مانند SSD که به داده‌های زیاد وابسته هستند.
  + کیفیت پایین تصاویر (مانند نویز، وضوح پایین یا پس‌زمینه‌های شلوغ) می‌تواند تشخیص اشیاء را دشوار کند.
  + تنوع ناکافی در داده‌ها (مانند زاویه‌های مختلف، شرایط نوری متفاوت یا اندازه‌های مختلف اشیاء) می‌تواند دقت مدل را کاهش دهد.
* **راهکارها:**
  + **افزایش داده‌ها:**
    - استفاده از تکنیک‌های افزایش داده (Data Augmentation) مانند چرخش، برش، تغییر مقیاس، و افزودن نویز برای ایجاد تصاویر جدید.
    - جمع‌آوری داده‌های بیشتر از شرایط واقعی.
  + **تمیز کردن داده‌ها:**
    - حذف داده‌های نویزی یا کم‌کیفیت.
  + **تنوع در داده‌ها:**
    - اطمینان از وجود تنوع در زاویه، اندازه، و شرایط نوری در تصاویر.

**2. اندازه اشیاء**

* **تأثیر:**
  + مدل‌ها معمولاً در تشخیص اشیاء کوچک ضعیف عمل می‌کنند. همان‌طور که در نتایج مشخص شد، SSD در تشخیص اشیاء کوچک عملکرد بسیار ضعیفی داشت.
  + اشیاء کوچک ممکن است در نقشه‌های ویژگی (Feature Maps) به وضوح دیده نشوند.
* **راهکارها:**
  + **استفاده از انکرهای کوچک‌تر:**
    - تنظیم انکر باکس‌ها برای پوشش بهتر اشیاء کوچک.
  + **مدل‌های چندرزولوشنه:**
    - استفاده از معماری‌هایی مانند Feature Pyramid Networks (FPN) یا EfficientDet که اطلاعات چندرزولوشنه را برای شناسایی اشیاء کوچک ترکیب می‌کنند.

**3. معماری مدل**

* **تأثیر:**
  + مدل‌های پیچیده‌تر مانند Faster R-CNN دقت بالاتری دارند، اما زمان محاسباتی بیشتری نیاز دارند.
  + مدل‌های سبک‌تر مانند SSD سرعت بیشتری دارند اما در دقت، به خصوص برای داده‌های کم، ضعیف‌تر عمل می‌کنند.
* **راهکارها:**
  + **استفاده از معماری‌های ترکیبی:**
    - معماری‌هایی مانند RetinaNet یا EfficientDet می‌توانند تعادلی میان دقت و سرعت ایجاد کنند.
  + **تنظیم معماری موجود:**
    - تغییر لایه‌های خروجی یا استفاده از انکرهای بیشتر در SSD برای بهبود دقت.

**4. بهینه‌سازی (Optimization)**

* **تأثیر:**
  + انتخاب نادرست بهینه‌ساز یا نرخ یادگیری می‌تواند منجر به یادگیری ناقص یا بیش‌برازش شود.
  + در مدل‌هایی مانند SSD، تنظیمات ضعیف می‌تواند بر تطبیق مدل با داده‌های کم تأثیر منفی بگذارد.
* **راهکارها:**
  + **بهینه‌ساز مناسب:**
    - استفاده از بهینه‌سازهایی مانند Adam یا AdamW که نرخ یادگیری تطبیقی دارند.
  + **تنظیم نرخ یادگیری:**
    - استفاده از Scheduler برای کاهش تدریجی نرخ یادگیری در طول آموزش.
  + **کاهش گرادیان‌های ناپایدار:**
    - استفاده از تکنیک‌های Clipping برای جلوگیری از انفجار گرادیان‌ها.

**5. تنظیمات هایپرپارامترها**

* **تأثیر:**
  + مقادیر پیش‌فرض هایپرپارامترها (مانند تعداد انکرها، نسبت ابعاد، یا نرخ یادگیری) ممکن است برای مسئله خاص شما مناسب نباشند.
* **راهکارها:**
  + **تنظیم انکرها:**
    - تغییر تعداد و نسبت ابعاد انکرها برای انطباق با اندازه و شکل اشیاء در داده‌ها.
  + **آزمایش نرخ یادگیری:**
    - آزمایش مقادیر مختلف نرخ یادگیری و وزن‌دهی (Weight Decay) برای یافتن مقدار بهینه.

**6. محدودیت‌های سخت‌افزاری**

* **تأثیر:**
  + مدل‌های پیچیده‌تر به سخت‌افزار قدرتمندتری نیاز دارند و ممکن است روی GPUهای ضعیف‌تر عملکرد مناسبی نداشته باشند.
* **راهکارها:**
  + **استفاده از مدل‌های سبک‌تر:**
    - برای محدودیت سخت‌افزاری می‌توان از مدل‌های سبک‌تر مانند MobileNet SSD استفاده کرد.
  + **استفاده از تسریع‌کننده‌ها:**
    - استفاده از TensorRT یا ONNX برای بهبود سرعت مدل.

**جمع‌بندی راهکارها برای بهبود دقت و سرعت:**

1. **برای بهبود دقت:**
   * افزایش داده‌های آموزشی با استفاده از Data Augmentation یا جمع‌آوری داده‌های بیشتر.
   * استفاده از معماری‌هایی با قابلیت تشخیص بهتر اشیاء کوچک مانند FPN.
   * تنظیم دقیق انکرها و نرخ یادگیری.
2. **برای بهبود سرعت:**
   * استفاده از مدل‌های سبک‌تر مانند MobileNet SSD.
   * کاهش اندازه تصاویر ورودی برای کاهش زمان پردازش.
   * استفاده از تسریع‌کننده‌های سخت‌افزاری مانند TensorRT.