به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سوم

نام و نام خانوادگی	**بهراد موسایی شیرمحمد - محمد جواد رنجبر کلهرودی**
شماره دانشجویی	**810101173 - 810101278**
تاریخ ارسال گزارش	**۱۴۰۲.۰۹.۱۷**

فهرست

پاسخ ۱. SAM^1

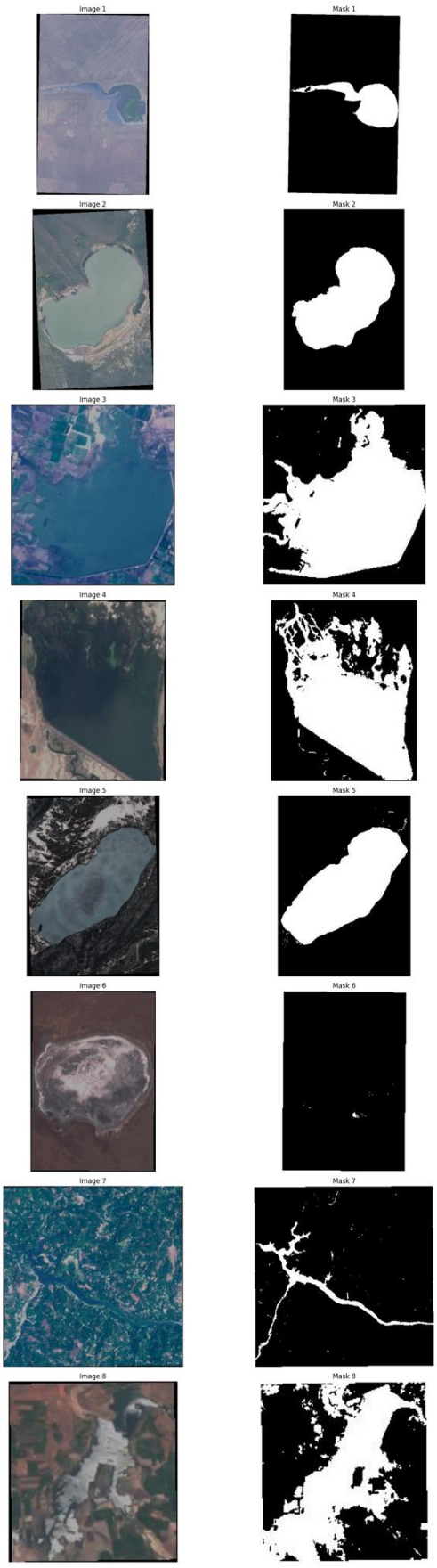
در این سوال تمام بخش‌های به همراه بخش امتیازی پاسخ داده شده است.

در این پرسش شما روی Fine-Tune کردن مدل جدید منتشر شده شرکت Meta برای مسئله Semantic Segmentation کار خواهید کرد تمرکز ،اصلی درک مفاهیم Segmentation تصویر، کار با مدل معرفی شده و تنظیم آن برای مسئله مورد نظر است.

حال پاسخ هر بخش به صورت زیر ارایه می شود:

۱ـ۱ آماده سازی مجموعه داده

در این بخش طبق درخواست های داده شده عمل می کنیم و پس از دانلود تصاویر, عکس اصلی را در کنار عکس ماسک قرار می دهیم و داده ها را ۹ به ۱ برای آموزش و تست تقسیم می کنیم.



شکل **۱**:نمونه تصاویر

توضیحات مجموعه داده:

مجموعه داده مورد استفاده برای این کار شامل تصاویر و ماسک های مربوطه است که در دایرکتوری های جداگانه (تصاویر و ماسک ها) ذخیره شده اند. مسیر فهرست مجموعه داده ها به صورت /content/Water Bodies Dataset/ مشخص می شود.

تقسیم داده ها:

تابع train_test_split از Scikit-learn برای تقسیم مجموعه داده ها به مجموعه های آموزشی و آزمایشی استفاده می شود. مسیرهای تصویر و ماسک به طور جداگانه بازیابی و مرتب شده اند. یک تقسیم ۹۰-۱۰ اعمال می شود که ۹۰٪ از داده ها را برای آموزش و ۱۰٪ را برای آزمایش اختصاص می دهد. فرآیند تقسیم توزیع متوازن داده ها را برای آموزش و ارزیابی تضمین می کند.

مولدهای داده:

ژنراتورها برای مجموعه های آموزشی و آزمایشی ایجاد شده اند تا بارگذاری و پیش پردازش داده ها را تسهیل کنند. این ژنراتورها از مسیرهای مشخص شده تصویر و برچسب همراه با یک عملکرد پردازنده برای کنترل افزایش داده ها (در صورت ارائه) استفاده می کنند.

ایجاد مجموعه داده TensorFlow:

تابع tf.data.Dataset.from_generator TensorFlow برای تبدیل داده ها از ژنراتورها به مجموعه داده های TensorFlow استفاده می شود.

راه اندازی خط لوله داده:

برای آموزش کارآمد، خط لوله داده با استفاده از عملکردهای مختلف TensorFlow بهینه شده است.

برای مجموعه داده آموزشی:

مخلوط کردن با اندازه بافر ۱ اعمال می شود. اندازه batch روی ۲ تنظیم شده است تا چندین نمونه را به طور همزمان پردازش کند. واکشی اولیه برای همپوشانی پیش پردازش داده ها و اجرای مدل اجرا می شود و عملکرد کلی را بهبود می بخشد.

برای مجموعه داده آزمایشی:

اندازه دسته به طور مشابه به مجموعه داده آموزشی مشخص می شود. پیش واکشی نیز برای کارایی در حین ارزیابی اعمال می شود. متغیر auto برای تعیین اندازه بافر در حین واکشی اولیه، بهینه سازی خط لوله بر اساس قابلیت های سیستم استفاده می شود. اندازه دسته و مقادیر بافر ترکیبی به ترتیب ۲ و ۱ انتخاب می شوند، اما بسته به منابع محاسباتی و نیازهای مدل می توان آنها را تنظیم کرد.

۱-۲ بارگذاری مدل

در این گزارش، ما به بررسی کدهای پایتون برای اجرای مدل Segment Anything Model (SAM) می‌پردازیم که با استفاده از کتابخانه‌های PyTorch و Hugging Face Transformers پیاده‌سازی شده است. این مدل برای کاربرد در تشخیص و جداسازی تصاویر به کمک فناوری پردازش تصویر و یادگیری ماشین طراحی شده است.

کدگذار تصویر (Image Encoder):

این کلاس با استفاده از یک مدل پیش‌آموزش داده شده از Vision Transformer (ViT) به عنوان اساس، تصاویر ورودی را به فضای ویژگی تبدیل می‌کند. این امکان را به مدل می‌دهد که ویژگی‌های مهم تصویر را شناسایی کرده و برای پردازش‌های بعدی آماده کند.

کدگذار پرامپت (Prompt Encoder):

این کلاس از کدگذار متنی CLIP که به صورت پیش‌آموزش داده شده در دسترس است، برای تبدیل پرامپت‌های متنی به نمایش‌های عددی استفاده می‌کند. این کدگذار به مدل این توانایی را می‌دهد که از پرامپت‌های متنی برای بهبود تشخیص و جداسازی استفاده کند.

کدگذار ماسک (Mask Decoder):

کدگذار ماسک، نمایش‌های عددی تولید شده توسط کدگذار تصویر و کدگذار پرامپت را گرفته و به ماسک‌های جداسازی تبدیل می‌کند. این بخش با استفاده از بلوک‌های دیکودر Transformer، تصاویر و پرامپت‌ها را با هم ترکیب کرده و ماسک‌های مورد نظر را تولید می‌کند.

مدل SAM:

کلاس SAM شامل کدگذار تصویر، کدگذار پرامپت، و کدگذار ماسک می‌باشد. این کلاس وظیفه اصلی پیش‌بینی ماسک‌ها برای تصاویر ورودی را بر عهده دارد. کلاس SAM به گونه‌ای طراحی شده است که می‌تواند به صورت موثر با پرامپت‌های مختلف کار کند و قابلیت استفاده در برنامه‌های کاربردی واقعی را دارد.

برای استفاده از این مدل، باید بارگذاری داده‌ها، تعریف توابع زیان، و حلقه‌های آموزشی را نیز پیاده‌سازی کرد. این بخش‌ها شامل بارگذاری داده‌های آموزشی و اعتبارسنجی، تعریف توابع زیان برای ارزیابی دقت پیش‌بینی‌های مدل و اجرای حلقه‌های آموزشی برای بهبود عملکرد مدل بر اساس داده‌ها می‌باشند.

البته این بخش های مختلف یک مدل sam است که ما به صورت مستقیم با آن ها برخورد نخواهیم داشت و از کتابخانه meta برای import کردن مول استفاده می کنیم.

برای ساده‌تر شدن و کمتر شدن حجم پردازشی مدل بخش‌های ذکر شده شامل vision encoder و prompt encoder را freeze می‌کنیم.

```
# initialize SAM model and optimizer
sam = TFSamModel.from_pretrained("facebook/sam-vit-base")
optimizer = keras.optimizers.Adam(1e-5)

for layer in sam.layers:
    if layer.name in ["vision_encoder", "prompt_encoder"]:
        layer.trainable = False
```

شکل **۲** فریز کردن بخش‌هایی از مدل

۱-۳. تقویت داده

در این بخش برای اجرای "تقویت داده" (Data Augmentation) بر روی مجموعه داده‌ای از تصاویر و ماسک‌های مربوطه در پردازش تصویر و بینایی ماشین استفاده می‌شود. کد از کتابخانه Albumentations برای اعمال تغییراتی مانند انعکاس افقی، تغییر روشنایی و کنتراست، چرخش، تغییر رنگ RGB و تاری بر روی تصاویر استفاده می‌کند. این تغییرات به منظور افزایش تنوع داده‌ها و کمک به مدل‌های یادگیری عمیق برای بهتر تعمیم دادن و بهبود عملکرد آنها در شرایط واقعی اعمال می‌شوند.

با استفاده از این کتابخانه می‌توان Augmentationها را روی هم ماسک هم داده‌های آموزش باهم اجرا کرد.

ابتدا، کد پوشه‌هایی که تصاویر و ماسک‌ها در آن قرار دارند را بارگذاری می‌کند و لیستی از فایل‌های موجود در این پوشه‌ها را ایجاد می‌کند. سپس، برای هر جفت تصویر و ماسک که نام یکسانی دارند، مسیرهای کامل را مشخص می‌کند و تصاویر را بارگذاری می‌کند. در این نقطه، کد با استفاده از ادعاها (assertions) اطمینان حاصل می‌کند که تصاویر به درستی به عنوان آرایه‌های نامپای (numpy arrays) بارگذاری شده‌اند که این فرمت ورودی مورد انتظار کتابخانه Albumentations است.

پس از آن، کد یک سری تغییرات تصادفی را بر روی هر تصویر و ماسک اعمال می‌کند و نتایج را نمایش می‌دهد. هر تغییرات به صورت تصادفی با توجه به احتمال مشخص شده برای هر عملیات اعمال می‌شود. این فرآیند برای تعداد مشخصی از دفعات تکرار می‌شود تا نمونه‌های متنوعی از داده‌های تقویت شده تولید شود.

تقویت هایی که انجام دادیم به شرح زیر خواهد بود:

۱. `A.HorizontalFlip (p=0.5)`: این دستور تصویر را با احتمال ۵۰٪ به صورت افقی (چپ به راست) وارونه می‌کند.

۲. `A.RandomBrightnessContrast (p=0.2)`: با احتمال ۲۰٪، درخشندگی و کنتراست تصویر را به صورت تصادفی تغییر می‌دهد. این به تنوع در نور و کنتراست تصاویر کمک می‌کند.

۳. `A.Rotate (limit=40, p=0.9)`: با احتمال ۹۰٪، تصویر را در یک زاویه تصادفی تا حداکثر ۴۰ درجه می‌چرخاند. این برای ایجاد تنوع در جهت گیری تصاویر است.

۴. `A.RGBShift (r_shift_limit=25, g_shift_limit=25, b_shift_limit=25)`: رنگ‌های قرمز، سبز و آبی (RGB) تصویر را تا حداکثر ۲۵ واحد به صورت جداگانه جابجا می‌کند. این تغییرات رنگی می‌تواند به مدل کمک کند تا بهتر با تغییرات رنگ در دنیای واقعی کنار بیاید.

۵. `A.Blur (blur_limit=3)`: تصویر را تا حداکثر محدودیت ۳ واحدی محو می‌کند. این برای شبیه‌سازی شرایطی است که در آن تصویر کمی تار است.

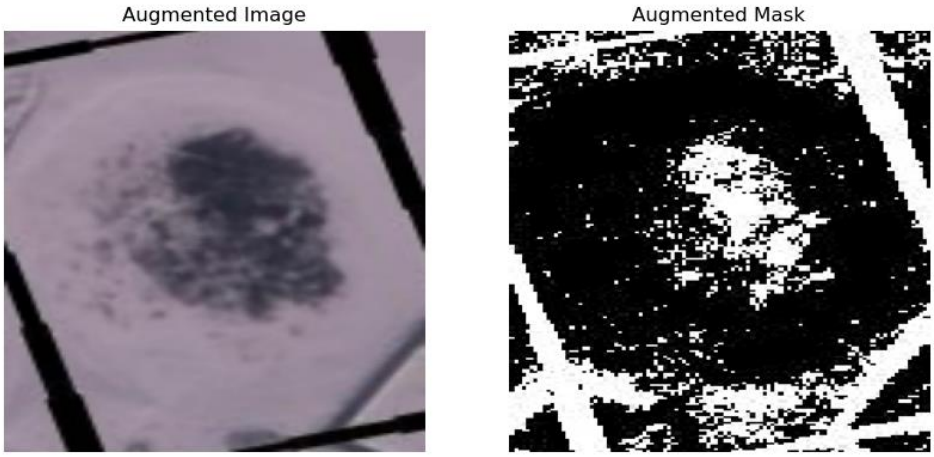
۶. `A.Resize (224, 224)`: اندازه تصویر را به ۲۲۴ در ۲۲۴ پیکسل تغییر می‌دهد. این معمولاً برای تطابق اندازه تصاویر با ورودی مورد نیاز شبکه‌های عصبی مصنوعی انجام می‌شود.

هدف از این افزایش داده‌ها ایجاد تنوع در داده‌های آموزشی است تا مدل‌های یادگیری ماشین بتوانند بهتر عمومیت یابند و در شرایط مختلف عملکرد بهتری داشته باشند. با استفاده از این روش‌ها داده‌های بیشتری تولید می‌کنیم که عمومیت‌پذیری مدل را چند برابر می‌کند تا برای داده‌هایی که مدل ندیده است بهتر عمل کند و عمومیت پذیری بالایی داشته باشد.

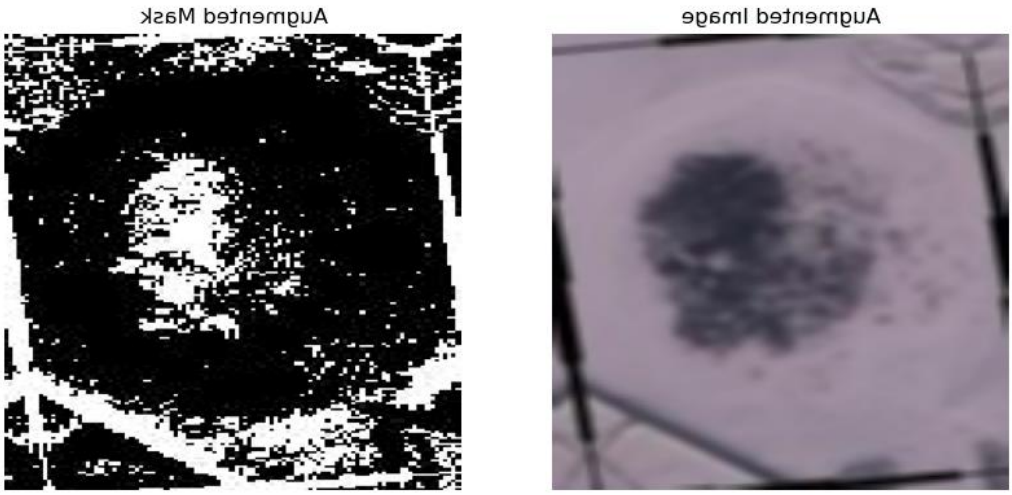
علاوه بر این روش‌های تقویتی استفاده شده توسط ما برای کار کرد این مجموعه داده مناسب است، و اطلاعات مفیدی را از بین نمی‌برد.

در نهایت، تصاویر تقویت شده به کمک کتابخانه matplotlib برای نمایش داده می‌شوند. این نمایش به تحلیلگر اجازه می‌دهد تا تأثیر تقویت داده را بصری بررسی کند و اطمینان حاصل کند که تغییرات مورد نظر به درستی بر روی داده‌ها اعمال شده‌اند.

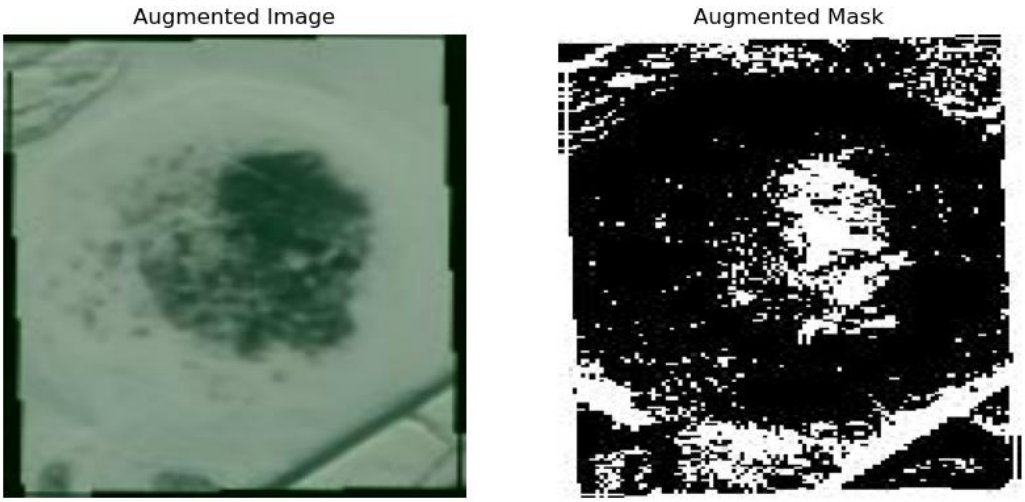
که نتایج به صورت زیر به دست می آید:



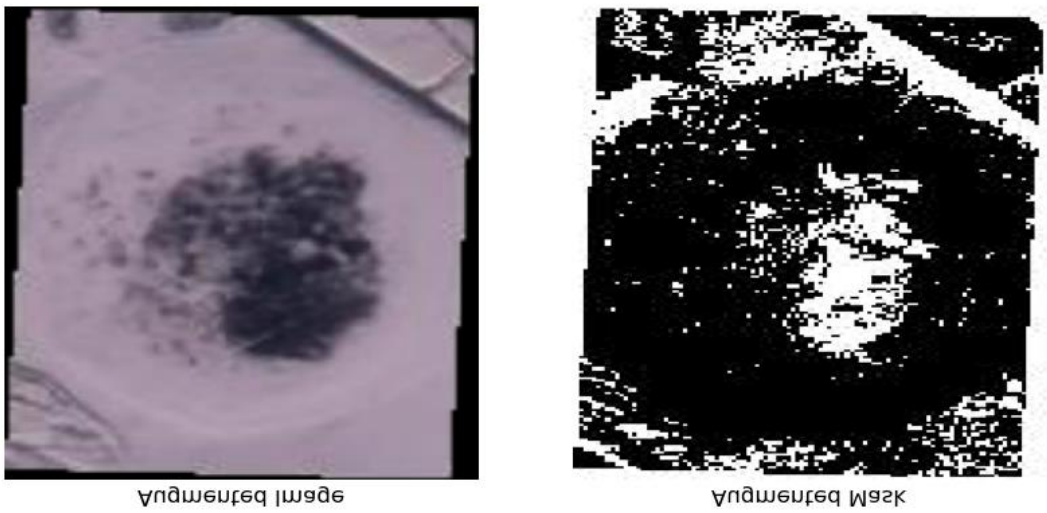
شکل **۳**: نتایج تقویت تصاویر ۱



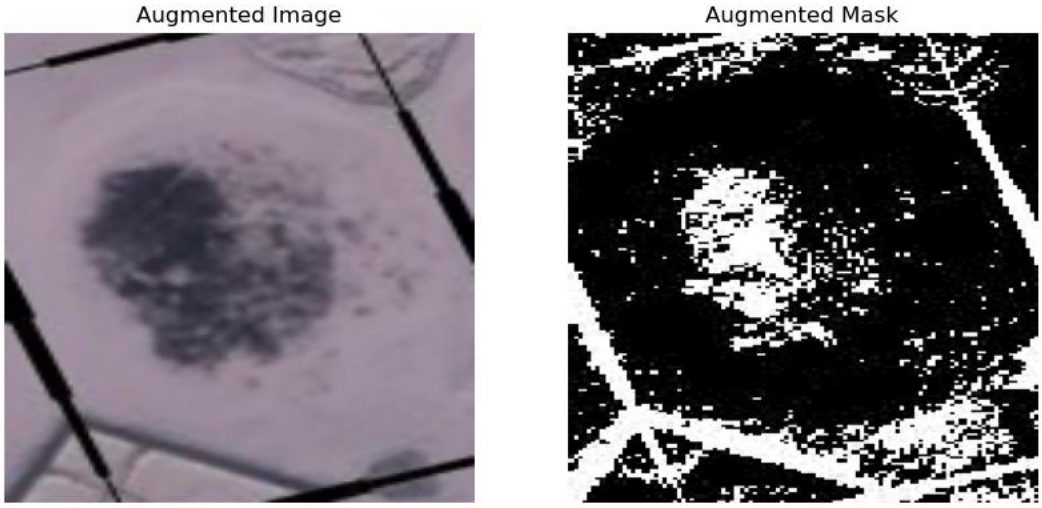
شکل **۴**:نتایج تقویت تصاویر ۲



شکل **۵**:نتایج تقویت تصاویر ۳



شکل ۶: نتایج تقویت تصاویر ۴



شکل ۷: نتایج تقویت تصاویر ۵

۱-۴. بهینه ساز متریک و تابع هزینه

۱. تابع `dice_coefficient`:

- این تابع مقدار شاخص Dice را محاسبه می‌کند، که یک معیار ارزیابی در پردازش تصویر و یادگیری ماشین است و برای مقایسه تشابه بین دو نمونه (معمولاً پیش‌بینی‌ها و برچسب‌های واقعی) استفاده می‌شود.

- `pred` و `target` دو تنسور هستند که پیش‌بینی‌های مدل و برچسب‌های واقعی را نشان مدهند.

- ابتدا، تقاطع (`intersection`) بین `pred` و `target` با ضرب درونی آن‌ها محاسبه می‌شود.

- سپس، اتحاد (`union`) با جمع کردن مجموع مقادیر در هر دو تنسور و کم کردن تقاطع به دست می‌آید.

تابع خطای Dice یک معیار متداول است که در ارزیابی عملکرد مدل‌های تقسیم تصویر استفاده می‌شود، به‌ویژه در شرایطی که داده‌ها نامتوازن هستند، مانند وقوع اشیاء کوچک در تصاویر. این تابع از ضریب Dice مشتق می‌شود، که یک معیار شباهت برای اندازه‌گیری توافق بین دو مجموعه است.

فرمول

خطای Dice با استفاده از فرمول زیر محاسبه می‌شود:

$$Dice\ error = 1 - \frac{2 * intersection + epsilon}{True + predicted + epsilon}$$

نقاطع: این متغیر میزان همپوشانی بین ماسک تقسیم‌بندی پیش‌بینی شده و ماسک واقعی را محاسبه می‌کند. این مقدار با جمع ضرب عنصر به عنصر ماسک پیش‌بینی شده و ماسک واقعی به دست می‌آید.

مقادیر واقعی و پیش‌بینی شده: این مقادیر مجموع پیکسل‌ها در ماسک واقعی و ماسک پیش‌بینی شده را نشان می‌دهند. این مقادیر به توان دو برده می‌شوند تا به مناطق همپوشانی بزرگتر توجه بیشتری شود.

اپسیلون (مقدار نرمال سازی): یک مقدار کوچک (معمولاً یک عدد مثبت بسیار کوچک) که به مخرج اضافه می‌شود تا تقسیم بر صفر جلوگیری شود. این مقدار استقرار عددی در هنگامی که مجموع واقعی و پیش‌بینی شده هر دو صفر باشند.

- خطای Dice تفاوت‌ها بین ماسک‌های تقسیم‌بندی پیش‌بینی شده و واقعی را مجازات می‌کند.

- هنگامی که ماسک پیش‌بینی شده کاملاً با ماسک واقعی همپوشانی دارد (همپوشانی کامل)، خطای Dice صفر است.

- با افزایش تفاوت بین ماسک‌ها، مقدار خطای Dice به یک نزدیک می‌شود.

- هدف در طول آموزش کمینه کردن خطای Dice است، که باعث می‌شود مدل به تولید ماسک‌های پیش‌بینی که با ماسک واقعی به خوبی همخوانی دارند، تشویق شود.

اهمیت:

- خطای Dice به‌ویژه زمانی مفید است که با مجموعه‌های نامتوازن سر و کار داریم که در آن کلاس‌های تقسیم‌بندی ناهموار توزیع شده‌اند.

- این معیار حساس به تغییرات کوچک در تقسیم‌بندی است و به همین دلیل در حالت‌هایی که تعریف دقیق اشیاء ضروری است، مانند تصاویر پزشکی، تشخیص اشیاء و زمینه‌های دیگر که تقسیم‌بندی دقیق ضروری است، مناسب است.

۲. تابع `iou:

- این تابع مقدار IoU (مخفف Intersection over Union) را محاسبه می‌کند، که یک معیار دیگر در پردازش تصویر و یادگیری ماشین برای مقایسه پوشش بین پیش‌بینی‌ها و برچسب‌های واقعی است.

- `pred` و `target` دوباره دو تنسور هستند که نشان‌دهنده پیش‌بینی‌ها و برچسب‌های واقعی هستند.

- تقاطع در اینجا مانند تابع قبلی محاسبه می‌شود.

- برای محاسبه اتحاد، ابتدا مجموع دو تنسور محاسبه شده و سپس تقاطع کم می‌شود.

- نهایتاً، نسبت تقاطع به اتحاد به عنوان مقدار IoU با فرمول intersection / union + 1e-5 محاسبه می‌شود.

تقاطع روی اتحاد (IoU) با استفاده از فرمول زیر محاسبه می شود:

$$IoU = \frac{Intersection}{Union}$$

ناحیه تقاطع: به ناحیه همپوشانی بین جعبه مرزی پیش بینی شده/ماسک قطعه بندی و جعبه/ماسک مرزی حقیقت زمین اشاره دارد.

Area of Union: نشان دهنده کل مساحت تحت پوشش هر دو جعبه/ماسک های مرزی حقیقت پیش بینی شده و زمینی است.

IoU همپوشانی بین مناطق حقیقت پیش بینی شده و زمینی را کمیت می کند.

از ۰ تا ۱ متغیر است، جایی که ۰ نشان دهنده عدم همپوشانی و ۱ نشان دهنده تطابق کامل بین مناطق حقیقت پیش بینی شده و زمین است.

مقادیر بالاتر IoU نشان دهنده تطابق بهتر بین مناطق حقیقت پیش بینی شده و زمینی است.

اهمیت:

IoU به طور گسترده در کارهایی مانند تشخیص اشیا و تقسیم بندی تصویر برای ارزیابی دقت پیش بینی های مدل استفاده می شود.

این به ویژه در سناریوهایی که مکان یابی یا ترسیم دقیق اشیاء ضروری است مفید است.

در طول آموزش، بهینه‌سازی برای مقادیر بالاتر IoU به مدل کمک می‌کند یاد بگیرد که جعبه‌های مرزبندی یا ماسک‌های تقسیم‌بندی دقیق‌تر و دقیق‌تر تولید کند.

این دو تابع در زمینه‌هایی مانند تشخیص اشیاء و تجزیه‌وتحلیل تصویر پزشکی برای ارزیابی دقت مدل‌ها بسیار کاربردی هستند.

۱-۵. Fine-Tune کردن مدل

کد گه در این بخش پیاده شده است , یک کلاس به نام `Generator` را تعریف می‌کند که برای پردازش تصاویر و ماسک‌های مرتبط با آن‌ها در زمینه یادگیری ماشین و تشخیص بصری استفاده می‌شود. این کلاس در زمینه‌هایی مانند تشخیص و جداسازی اشیاء در تصاویر (image segmentation) کاربرد دارد، به خصوص برای تنظیم دقیق مدل‌های پیشرفته مانند SAM (Segment Aware Models).

اجزای اصلی و عملکرد کلیدی کد عبارتند از:

۱. آماده‌سازی داده‌ها: کلاس ابتدا مسیرهای تصویر و ماسک را به عنوان ورودی دریافت می‌کند. این تصاویر و ماسک‌ها سپس برای پردازش بیشتر تغییر اندازه داده می‌شوند تا به یک اندازه استاندارد مثلا ۲۵۶ در ۲۵۶ برسند.

۲. اعمال افزایش داده (Augmentation): اگر افزایش داده تعریف شده باشد، این فرآیند به تصویر و ماسک اعمال می‌شود. این کار می‌تواند شامل تغییراتی مانند چرخش، تغییر رنگ، و یا تغییرات دیگر باشد تا به مدل کمک کند تا بر روی داده‌های متنوع‌تری آموزش ببیند و عمومی‌تر عمل کند.

۳. تولید کادر محدوده (Bounding Box): کلاس یک کادر محدوده بر اساس ماسک زمینه حقیقت (ground truth mask) تولید می‌کند. این کادر محدوده برای تعیین موقعیت و اندازه اشیاء در تصویر استفاده می‌شود.

۴. آماده‌سازی داده‌ها برای مدل: پس از اینکه تصویر و ماسک پردازش شدند، این کلاس اطلاعات لازم را برای ورودی مدل هوش مصنوعی آماده می‌کند. این شامل تبدیل تصویر و ماسک به فرمت مناسب و تولید متغیرهای ورودی مورد نیاز برای مدل است.

کلاس `Generator` به عنوان یک ابزار پیشرفته برای آماده‌سازی داده‌های تصویری عمل می‌کند و به افزایش کارایی و دقت مدل‌های تشخیص تصویر کمک می‌کند، به ویژه در زمینه‌هایی که نیاز به داده‌های دقیق و خوب تعریف شده برای آموزش و تنظیم دقیق مدل‌ها وجود دارد.

۱-۶. ارزیابی نتایج

در ابتدا با رویکر زیر داده ها را برای آموزش و تست آماده می کنیم:

این بخش از کد برای آماده‌سازی و سازمان‌دهی داده‌های تصویری و ماسک‌های مربوطه به منظور آموزش و ارزیابی مدل‌های یادگیری ماشین در زمینه تشخیص تصویر استفاده می‌شود. این کد به شرح زیر عمل می‌کند:

۱. دسته‌بندی داده‌ها: ابتدا مسیرهای تصاویر و ماسک‌های مرتبط با آن‌ها از داخل پوشه‌های مشخصی در دیتاست خوانده می‌شوند و سپس به ترتیب مرتب می‌شوند.

۲. تقسیم داده‌ها به دو بخش آموزش و تست: داده‌ها به دو بخش آموزشی و تست تقسیم می‌شوند که این کار با استفاده از تابع `train_test_split` از کتابخانه `sklearn` انجام می‌گیرد. در این مثال، ۱۰٪ از داده‌ها برای تست و ۹۰٪ برای آموزش استفاده می‌شود.

۳. ایجاد مولدهای داده (Generators): برای هر بخش (آموزش و تست)، یک مولد داده با استفاده از کلاس `Generator` که قبلا توضیح داده شد، ایجاد می‌شود. برای مجموعه آموزش، افزایش داده (augmentation) نیز اعمال می‌شود.

۴. تبدیل به دیتاست‌های TensorFlow: مولدهای داده به دیتاست‌های TensorFlow تبدیل می‌شوند تا به راحتی در مدل‌های یادگیری ماشین قابل استفاده باشند.

۵. پیکربندی دیتاست‌ها: دیتاست‌ها برای بهینه‌سازی عملکرد پردازش و آموزش، با استفاده از شافل (shuffle)، دسته‌بندی (batching)، و پیش‌خوانی (prefetching) پیکربندی می‌شوند. این تنظیمات به بهبود کارایی و سرعت پردازش داده‌ها کمک می‌کنند.

کلیت این کد به منظور سازمان‌دهی و بهینه‌سازی فرآیند آموزش و تست مدل‌های تشخیص تصویر با استفاده از دیتاست‌های تصویری و ماسک‌های مربوطه طراحی شده است. این به طور خاص برای استفاده در محیط‌های TensorFlow و یادگیری ماشین بهینه‌سازی شده است.

حال می خواهیم مدل Sam را بر روی این داده های آماده شده آموزش دهیم کهبه صورت زیر پیاده سازی می کنیم:

این کد شامل توابع و یک تابع آموزشی (`train_step`) برای آموزش یک مدل هوش مصنوعی در زمینه تشخیص تصویر (به خصوص برای مدل‌هایی که از تکنیک‌هایی مانند SAM استفاده می‌کنند) است. این کد به طور خاص برای ارزیابی و بهبود عملکرد مدل در تشخیص اشیاء در

تصاویر با استفاده از معیارهایی مانند IOU و Dice Coefficient طراحی شده است. اجزای اصلی کد عبارتند از:

۱. calculate_iou: تابعی برای محاسبه معیار IOU (Intersection Over Union) بین پیش‌بینی مدل و ماسک زمینه حقیقت (target). این معیار نشان می‌دهد که چقدر پیش‌بینی مدل با واقعیت تطابق دارد.

۲. calculate_dice_coefficient: تابعی برای محاسبه ضریب Dice، که یک معیار دیگر برای ارزیابی تطابق بین پیش‌بینی مدل و ماسک زمینه حقیقت است. این معیار به ویژه در تشخیص تصویر مورد استفاده قرار می‌گیرد.

۳. train_step: این تابع آموزشی برای انجام یک گام در فرآیند آموزش مدل استفاده می‌شود. ابتدا ورودی‌ها به مدل داده می‌شوند و خروجی‌ها (پیش‌بینی‌های مدل) بدست می‌آیند. سپس، با استفاده از تابع ضرر (loss function) که در اینجا `dice_loss` فرض شده است، ضرر محاسبه می‌شود. همچنین، معیارهای IOU و Dice Coefficient برای هر پیش‌بینی محاسبه می‌شوند. در نهایت، با استفاده از روش backpropagation و بهینه‌ساز (optimizer)، پارامترهای مدل به روز رسانی می‌شوند.

کلیت این کد به منظور آموزش و بهبود مدل‌های تشخیص تصویر با استفاده از تکنیک‌های پیشرفته و معیارهای ارزیابی دقیق طراحی شده است. این به مدل کمک می‌کند تا بهتر بتواند اشیاء را در تصاویر تشخیص دهد و دقت خود را در طول زمان بهبود ببخشد.

در ادامه برای نمایش ماسک های پیشبینی شده به صورت زیر عمل می کنیم:

این کد یک تابع به نام `display_images_with_masks` را تعریف می‌کند که برای نمایش تصاویر واقعی، ماسک‌های صحیح (درست) و ماسک‌های پیش‌بینی شده توسط یک مدل تشخیص تصویر مانند SAM برای چند نمونه تصادفی استفاده می‌شود. کارکرد کلی تابع به این صورت است:

۱. انتخاب تصادفی تصاویر: ابتدا تابع تعداد مشخصی (معمولا ۵) از شاخص‌ها را به صورت تصادفی از میان مسیرهای تصویر موجود انتخاب می‌کند.

۲. بارگذاری و نمایش تصاویر و ماسک‌ها: برای هر شاخص انتخاب شده، تصویر مربوطه و ماسک صحیح آن بارگذاری و به اندازه مورد نیاز (۲۵۵ در ۲۵۵ پیکسل) تغییر اندازه می‌دهد.

۳. پردازش تصویر و استنتاج: هر تصویر با استفاده از یک پردازشگر (مانند تابع `processor`) پردازش می‌شود و سپس به مدل SAM ارائه می‌شود تا ماسک‌های پیش‌بینی شده و امتیازات IOU آن‌ها تولید شود.

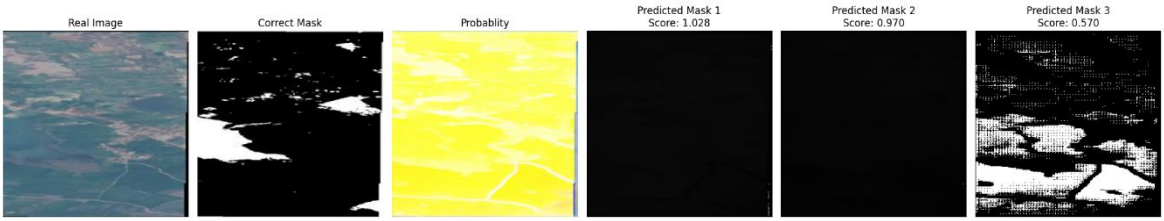
۴.نمایش نتایج: تابع سپس تصویر واقعی، ماسک صحیح و ماسک‌های پیش‌بینی شده را در کنار هم نمایش می‌دهد. هر ماسک پیش‌بینی شده با امتیاز IOU مرتبط با آن نمایش داده می‌شود تا کاربر بتواند دقت پیش‌بینی‌ها را ارزیابی کند.

این تابع به ویژه برای ارزیابی کیفیت و دقت مدل‌های تشخیص تصویر مانند SAM در تشخیص و جداسازی اشیاء در تصاویر استفاده می‌شود. با نمایش تصاویر واقعی، ماسک‌های صحیح و ماسک‌های پیش‌بینی شده، این تابع امکان مقایسه مستقیم بین نتایج واقعی و پیش‌بینی‌های مدل را فراهم می‌کند.

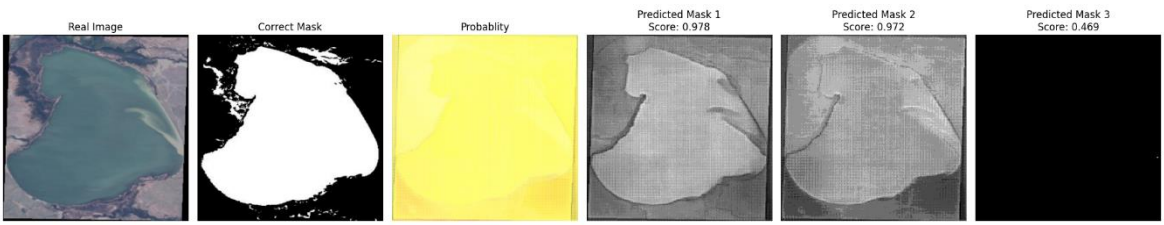
مدل را برای یک دوره fine tune می‌کنیم، نتایج برای داده‌های ارزیابی به صورت زیر می‌باشد:

```
Validation - Epoch 1: Average Loss = 0.9918, Average IoU = 0.4098,
                                    Average Dice Coefficient = 0.5654
```

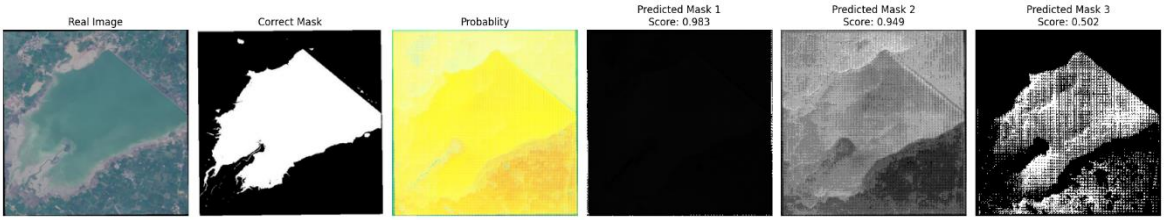
همچنین برای ۵ نمونه از داده‌ها نتایج به شکل زیر می‌باشد:



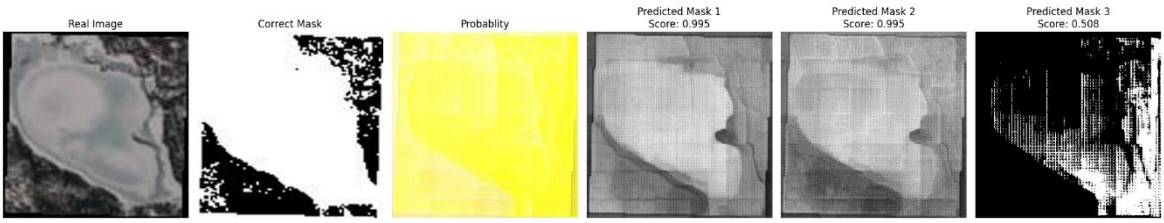
شکل ۸: نمایش نتایج ۱



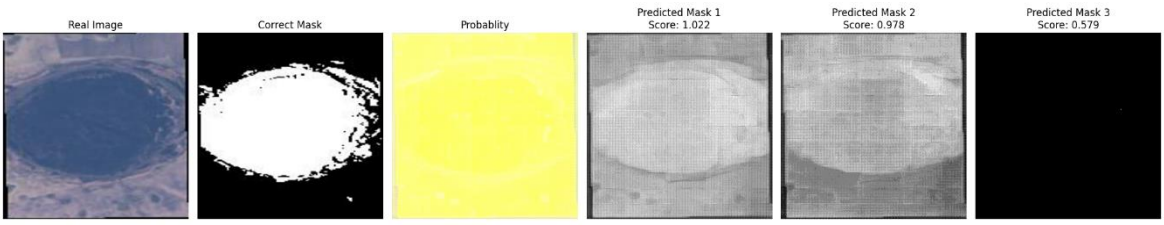
شکل ۹: نمایش نتایج ۲



شکل ۱۰ ماسک نمونه



شکل ۱۱ ماسک



شکل ۱۲ ماسک نمونه

پاسخ ۲ – آشنایی و پیاده سازی مدل Faster RCNN

در این سوال تمام بخش‌ها پاسخ داده شده‌اند.

در این بخش میخواهیم تا یک مدل Faster RCNN را با یک دیتاست آموزش دهیم و نتایج را بررسی کنیم برای این سوال مقاله Analysis of Object Detection Performance Based on Faster CNN را شبیه سازی میکنیم در مقاله برای آموزش از دیتاست Pascal VOC استفاده شده است. این دیتاست ۲۰ نوع کلاس و بیش از ده هزار عکس و annotation دارد بنابراین با توجه به محدودیت های موجود، میخواهیم نتایج این شبیه سازی را برای یک دیتاست کوچکتر بررسی کنیم میتوانید به دیتاست مورد استفاده در این مساله با این لینک دسترسی پیدا کنید در صورتی که تمایل به استفاده از پارامترهای متفاوتی از موارد گفته شده در مقاله ،دارد لطفا توجیه خود را در گزارش ذکر کنید.

۱-۲. توضیحات مدلها

برای درک بهتر تفاوت‌ها و بهبودهایی که در مدل Faster R-CNN نسبت به CNN و Fast R-CNN اعمال شده است، ابتدا به توضیح مختصر هر یک می‌پردازیم و سپس بهبودهای اعمال شده در Faster R-CNN را بررسی می‌کنیم. همچنین به کاربرد هر بخش از شبکه نیز خواهیم پرداخت.

(CNN) Convolutional Neural Network

CNN، که مخفف Convolutional Neural Network است، یک نوع خاص از شبکه‌های عصبی عمیق است که بیشتر برای پردازش داده‌هایی با ساختار شبکه‌ای مانند تصاویر استفاده می‌شود. این شبکه‌ها برای تشخیص و دسته‌بندی اشیاء در تصاویر، تحلیل ویدیوها، پردازش زبان طبیعی و سایر کاربردهای مشابه به کار می‌روند. در ادامه، جنبه‌های اصلی و کارکردهای کلیدی CNNها را بررسی می‌کنیم:

مؤلفه‌های اصلی CNN لایه‌های کانولوشن (Convolution Layers): این لایه‌ها ویژگی‌های اصلی تصاویر مانند لبه‌ها، گوشه‌ها، و بافت‌ها را استخراج می‌کنند. این کار با اعمال فیلترهایی که روی تصویر حرکت می‌کنند و واکنش‌هایی را در مکان‌هایی که ویژگی‌های خاصی را شناسایی می‌کنند، ایجاد می‌کنند.

لایه‌های فعال‌سازی: معمولاً پس از هر لایه کانولوشن، یک لایه فعال‌سازی قرار دارد. تابع فعال‌سازی معمول در CNNها تابع ReLU یا مشتقات آن است که به مدل کمک می‌کند تا ویژگی‌های غیرخطی را یاد بگیرد.

لایه‌های Pooling Layers : این لایه‌ها برای کاهش ابعاد فضایی (طول و عرض) ویژگی‌های استخراج شده بدون از دست دادن اطلاعات مهم به کار می‌روند. این کار باعث کاهش پیچیدگی محاسباتی مدل می‌شود.

لایه‌های Fully Connected: در انتهای شبکه، لایه‌هایی وجود دارند که هر نورون به تمام نورون‌های لایه قبل متصل است. این لایه‌ها به مدل کمک می‌کنند تا ویژگی‌های استخراج شده را برای تشخیص یا دسته‌بندی نهایی ترکیب کنند.

- مفهوم اصلی: CNN یک شبکه عصبی عمیق است که برای پردازش تصاویر به کار می‌رود.
- کاربرد: استخراج ویژگی‌ها از تصاویر از طریق لایه‌های کانولوشن (Conv Layer).

Fast R-CNN

Fast R-CNN یک مدل پیشرفته در زمینه تشخیص اشیاء در تصاویر است که به منظور بهبود کارایی و دقت نسبت به مدل‌های قبلی مانند R-CNN و SPPnet طراحی شده است. این مدل از چندین جنبه کلیدی برای افزایش سرعت و کاهش پیچیدگی محاسباتی بهره می‌برد. در ادامه، اجزای اصلی و ویژگی‌های کلیدی Fast R-CNN را توضیح می‌دهیم:

ویژگی‌های اصلی Fast R-CNN

۱. استفاده از شبکه‌های عصبی کانولوشنی (CNN) برای استخراج ویژگی‌ها: در Fast R-CNN، تمام تصویر یا یک منطقه معین از آن ابتدا از طریق یک شبکه کانولوشنی عمیق (مانند VGG16 یا AlexNet) عبور داده می‌شود تا ویژگی‌های تصویر استخراج شوند. این فرآیند باعث می‌شود تا استخراج ویژگی‌ها نسبت به مدل‌های قدیمی‌تر مانند R-CNN سریع‌تر و کارآمدتر باشد.

۲. Region of Interest (RoI) Pooling: پس از استخراج ویژگی‌ها، Fast R-CNN از یک رویکرد به نام RoI Pooling برای تبدیل ویژگی‌های استخراج شده از هر منطقه پیشنهادی (proposal region) به یک اندازه ثابت استفاده می‌کند. این کار امکان استفاده از شبکه‌های کاملاً متصل (fully connected layers) را برای هر منطقه پیشنهادی فراهم می‌کند.

۳. یکپارچگی تشخیص و دسته‌بندی: در مقابل R-CNN که ابتدا مناطق پیشنهادی را از تصویر جدا می‌کرد و سپس هر یک را به طور جداگانه پردازش می‌کرد، Fast R-CNN این فرآیندها را در یک شبکه یکپارچه ادغام می‌کند. این امر باعث کاهش قابل توجه زمان پردازش و بهبود کارایی می‌شود.

۴. loss function: Fast R-CNN از یک تابع ضرر چند وظیفه‌ای استفاده می‌کند که همزمان هم تشخیص (دسته‌بندی) اشیاء و هم رگرسیون محدوده (bounding box regression) را انجام می‌دهد. این رویکرد به مدل اجازه می‌دهد تا هم دسته‌بندی اشیاء در تصویر و هم تعیین موقعیت دقیق آن‌ها را بهینه کند.

کاربردهای Fast R-CNN

Fast R-CNN به طور گسترده‌ای در برنامه‌های تشخیص اشیاء مورد استفاده قرار می‌گیرد، جایی که نیاز به سرعت بالا و دقت قابل اعتماد است. این مدل در مقایسه با مدل‌های قدیمی‌تر، کارایی بهتری در تشخیص اشیاء مختلف در تصاویر دارد و می‌تواند در زمینه‌هایی مانند نظارت تصویری، خودروهای خودران، و سیستم‌های تحلیل تصویر پزشکی به کار رود.

در مجموع، Fast R-CNN یک گام مهم در پیشرفت تکنولوژی تشخیص اشیاء بود و مبنایی برای توسعه مدل‌های بعدی مانند Faster R-CNN و YOLO فراهم کرد.

- بهبود نسبت به CNN: این مدل نسبت به CNNهای سنتی در تشخیص اشیاء و دقت بالاتری دارد.
- کاربرد اصلی: تشخیص اشیاء در تصاویر با استفاده از روش‌های بهینه‌تر.
- مؤلفه‌های کلیدی: شامل ROI Pooling و لایه‌های کانولوشن و دسته‌بندی (Classification).

Faster R-CNN

Faster R-CNN یک معماری پیشرفته در زمینه تشخیص اشیاء است که برای رفع محدودیت‌های مدل‌های قبلی مانند Fast R-CNN و R-CNN طراحی شده است. این مدل از چندین نوآوری کلیدی

برای بهبود سرعت و دقت در تشخیص اشیاء در تصاویر استفاده می‌کند. در زیر، اجزای اصلی و ویژگی‌های کلیدی Faster R-CNN را توضیح می‌دهیم:

ویژگی‌های اصلی Faster R-CNN

۱. Region Proposal Network (RPN): یکی از بزرگترین بهبودها در Faster R-CNN اضافه کردن RPN است. این شبکه وظیفه تولید مناطق پیشنهادی (region proposals) را دارد که ممکن است اشیاء در آن‌ها قرار داشته باشند. RPN مستقیماً بر روی ویژگی‌های استخراج شده توسط شبکه کانولوشنی عمل می‌کند و مناطق پیشنهادی را با سرعت و دقت بالا تولید می‌کند. شبکه ای که مناطق شیء کاندید (جعبه های مرزی) را در نقشه های ویژگی ایجاد شده توسط شبکه ستون فقرات پیشنهاد می کند. با لغزش یک شبکه کوچک (معمولاً چند لایه کانولوشنال) بر روی نقشه های ویژگی، پیش بینی امتیازات شیئی و مختصات جعبه محدود، مناطق بالقوه برای اشیاء را پیشنهاد می کند.

۲. یکپارچگی با شبکه کانولوشنی: در Faster R-CNN، RPN و شبکه کانولوشنی برای استخراج ویژگی‌ها به صورت یکپارچه عمل می‌کنند. این یکپارچگی باعث می‌شود که تولید مناطق پیشنهادی و استخراج ویژگی‌ها با سرعت بیشتری انجام شود.

۳. RoI Pooling: مانند Fast R-CNN، Faster R-CNN نیز از RoI Pooling برای تبدیل ویژگی‌های استخراج شده از هر منطقه پیشنهادی به یک اندازه ثابت استفاده می‌کند. این کار به مدل اجازه می‌دهد تا از لایه‌های کاملاً متصل برای تشخیص و دسته‌بندی اشیاء استفاده کند. هنگامی که مناطق توسط RPN پیشنهاد می‌شوند، ادغام RoI یا تراز RoI برای تغییر اندازه و استخراج نقشه‌های ویژگی با اندازه ثابت از این مناطق استفاده می‌شود و به لایه‌های بعدی اجازه می‌دهد با اندازه‌های ورودی ثابت کار کنند.

۴. جعبه های لنگر:

مجموعه ای از جعبه های لنگر از پیش تعریف شده با مقیاس ها و نسبت های مختلف که توسط RPN برای پیشنهاد مناطق استفاده می شود.

۵. طبقه بندی کننده و رگرسیون:

Classifier: محتوای هر منطقه پیشنهادی را به کلاس های شی یا پس زمینه مختلف طبقه بندی می کند.

Regressor: مختصات جعبه مرزی پیشنهادی را برای تطبیق بهتر شی در منطقه اصلاح می کند.

۶. NMS

عملکرد Non-Maximum Suppression (NMS) یک تکنیک پس‌پردازش معمول در وظایف شناسایی اشیاء است که شامل روش‌هایی مانند Faster R-CNN، YOLO و SSD است.

هدف این تکنیک فیلتر و بهبود خروجی الگوریتم‌های شناسایی اشیاء با کاهش جعبه‌های مربوط به اشیاء تشخیص داده شده تکراری یا همپوشانی دارد.

۱. خروجی شناسایی اشیاء:

- الگوریتم‌های شناسایی اشیاء اغلب چندین جعبه‌ی مربوط به همان اشیا را به دلیل عوامل مختلفی مانند اندازه‌ها، موقعیت‌ها یا همپوشانی‌های فیلدهای رسیدگی متفاوت ایجاد می‌کنند.

۲. امتیازدهی به شناسایی‌ها:

- هر جعبه مربوط به یک اشیا با یک امتیاز اعتماد مرتبط است که نشان‌دهنده احتمال این است که جعبه شامل یک اشیا مورد نظر است. این امتیاز معمولاً بر اساس احتمال خروجی طبقه‌بندی‌کننده محاسبه می‌شود.

۳. فرآیند NMS:

- برای هر کلاس به صورت مستقل (در صورت تشخیص چند کلاس):

- جعبه‌های شناسایی شده را بر اساس امتیاز اعتماد به صورت نزولی مرتب می‌کنیم.

- جعبه‌ای که بیشترین امتیاز اعتماد را دارد را انتخاب کرده و آن را به عنوان 'کاندید شناسایی' در نظر می‌گیریم.

- این جعبه کاندید را با سایر جعبه‌ها مقایسه می‌کنیم.

- برای هر جعبه بعدی:

- اشتراک بیشینه به اتحاد (IoU) با جعبه کاندید را محاسبه می‌کنیم.

- اگر IoU بیشتر از یک آستانه خاص باشد (معمولاً حدود ۰/۵ یا بالاتر)، آن جعبه را که احتمالاً به همان اشیا کاندید مرتبط است را کنترل می‌کنیم و حذف می‌کنیم.

- به جعبه با بیشترین امتیاز بعدی می‌رویم و فرآیند را تکرار می‌کنیم تا همه جعبه‌ها پردازش شوند.

۴. خروجی نهایی:

- جعبه‌های باقی‌مانده پس از این فرآیند، جعبه‌های شناسایی نهایی محسوب می‌شوند.

NMS کمک می‌کند تا جعبه‌های شناسایی تکراری حذف شده و فقط جعبه‌هایی که بیشترین اعتماد و همپوشانی کمتری دارند باقی بمانند. این یک گام اساسی در بهبود نتایج شناسایی اشیاء است، تضمین می‌کند که خروجی نهایی اطلاعات دقیق و مختصری در مورد اشیا شناسایی شده ارائه کند.

کاربردهای Faster R-CNN

Faster R-CNN به دلیل سرعت و دقت بالای خود در تشخیص اشیاء، در زمینه‌های متعددی مانند نظارت تصویری، خودروهای خودران، تحلیل تصاویر پزشکی و سیستم‌های تشخیص چهره کاربرد دارد. این مدل یکی از محبوب‌ترین مدل‌ها در حوزه تشخیص اشیاء است و به عنوان مبنایی برای توسعه مدل‌های پیشرفته‌تر در این زمینه به کار می‌رود.

در مجموع، Faster R-CNN با ادغام RPN در معماری خود و یکپارچه‌سازی فرآیندهای مختلف، تحولی بزرگ در زمینه تشخیص اشیاء ایجاد کرده است. این مدل توانسته است تعادلی بین سرعت و دقت ایجاد کند و به این ترتیب، کاربردهای گسترده‌ای در صنایع مختلف پیدا کرده است.

- بهبود نسبت به Fast R-CNN و CNN:

- Region Proposal Network (RPN): برای پیش‌بینی مناطق مورد علاقه در تصویر. این بهبود باعث کاهش نیاز به منابع خارجی برای پیشنهاد مناطق می‌شود.

- سرعت بالاتر و دقت بهتر: به لطف RPN، Faster R-CNN سریع‌تر و دقیق‌تر از Fast R-CNN عمل می‌کند.

به طور خلاصه، Faster R-CNN با ادغام RPN در معماری خود، فرایند تشخیص اشیاء را سریع‌تر و دقیق‌تر نسبت به مدل‌های قبلی مانند Fast R-CNN و سنتی‌ترین CNNها انجام می‌دهد. این بهبودها شامل کارایی بهتر در پیش‌بینی مناطق مورد علاقه و کاهش نیاز به منابع خارجی برای این کار می‌شود.

۲-۲. پیش پردازش

این کد پایتون مراحل استخراج تصاویر و برچسب‌های آن‌ها از یک فایل فشرده، نمایش، تغییر اندازه و تنظیم مجدد مختصات جعبه‌های کادربندی (Bounding Box) را انجام می‌دهد. در ادامه، توضیح کامل این کد به فارسی ارائه شده است:

۱. وارد کردن کتابخانه‌های مورد نیاز:

- `zipfile`: برای کار با فایل‌های فشرده.
- `os`: برای کار با سیستم فایل‌های عامل.
- `OpenCV (cv2)`: برای کار با تصاویر.
- `xml.etree.ElementTree`: برای تجزیه فایل‌های XML.
- `matplotlib.pyplot`: برای نمایش تصاویر.

۲. تابع `parse_xml`: این تابع فایل XML مرتبط با هر تصویر را تجزیه کرده و مختصات جعبه‌های کادربندی موجود در آن را استخراج می‌کند.

۳. استخراج فایل فشرده: با استفاده از کتابخانه `zipfile`، فایل فشرده محتوی تصاویر و فایل‌های XML استخراج می‌شود.

۴. تعیین مسیر پوشه 'train': مسیر پوشه حاوی تصاویر آموزشی تعیین می‌شود.

۵. انتخاب ۵ تصویر و فایل‌های XML مربوطه: از بین تمام فایل‌های موجود در پوشه "train"، ۵ تصویر و فایل‌های XML متناظر با آن‌ها انتخاب می‌شوند.

۶. بارگذاری تصاویر و جعبه‌های کادربندی: هر تصویر به همراه جعبه‌های کادربندی مربوطه بارگذاری می‌شوند.

۷. تغییر اندازه تصاویر و جعبه‌های کادربندی: تصاویر به اندازه یکسانی (x256۲۵۶ پیکسل) تغییر اندازه داده می‌شوند. همچنین مختصات جعبه‌های کادربندی متناسب با اندازه جدید تصویر تنظیم مجدد می‌شوند.

۸. نمایش تصاویر با اندازه تغییر یافته و جعبه‌های کادربندی تنظیم شده: تصاویر با اندازه جدید و جعبه‌های کادربندی متناظر با آن‌ها نمایش داده می‌شوند. جعبه‌های کادربندی با خطوط سبز رنگ مشخص شده‌اند.

این کد برای پردازش مجموعه داده‌های تصویری که شامل برچسب‌های مکانی (مانند جعبه‌های کادربندی) هستند مناسب است. این نوع پردازش در وظایف یادگیری ماشین مانند تشخیص اشیاء کاربرد دارد.

در نهایت این ۵ تصویر به صورت زیر ارایه می شوند:

شکل ۱۳: ۵ تصویر اولیه

شکل ۱۴: ۵ تصویر بعد از تغییر سایز

حال تمام موارد گفته شده برای faster rcnn را به ترتیب پیاده‌سازی می‌کنیم:

ابتدا داده‌ها را از فولدر مربوطه می‌خوانیم و سایز آن‌ها را تغییر می‌دهیم.

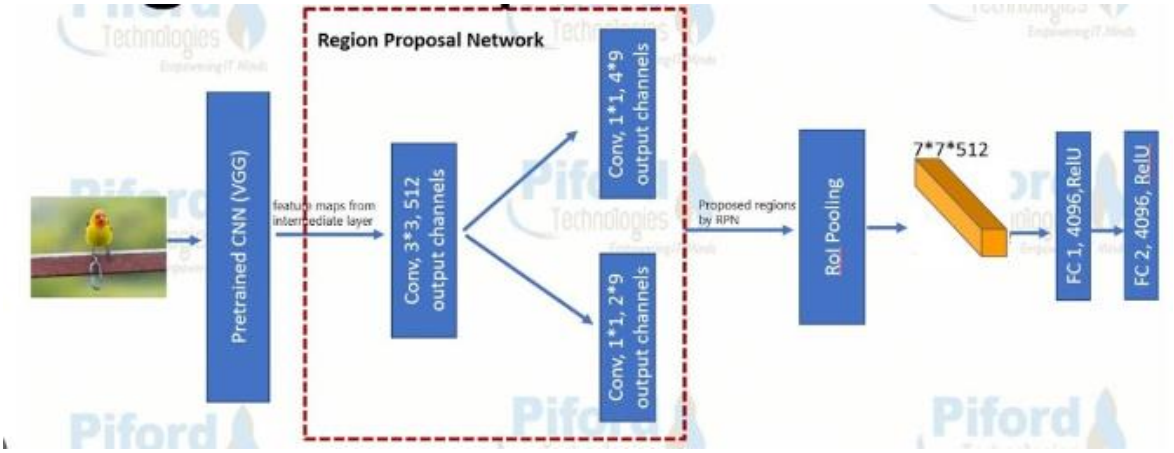
برای اینکه boundingboxها نیز دچار مشکل نشوند، تغییرات مربوط به سایز را برای آن‌ها نیز اعمال می‌کنیم، حال بخش‌های مختلف شبکه را پیاده سازی می‌کنیم:

Conv Layer:

برای ساده‌سازی از backbone مدل VGG برای پیاده‌سازی این شبکه استفاده می‌کنیم.

Region proposal network:

این مدل نیز با توجه به معماری مربوط به آن پیاده‌سازی می‌کنیم:



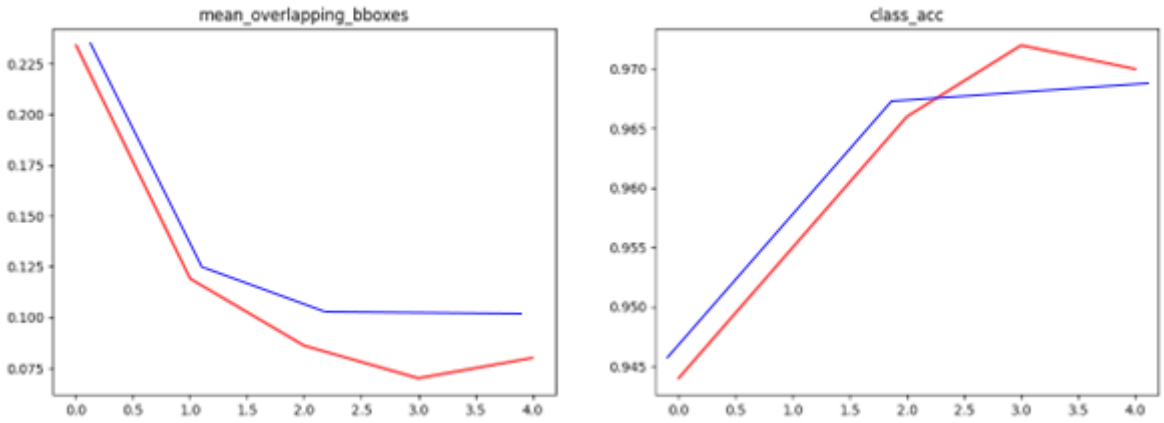
شکل ۱۵ **RPN**

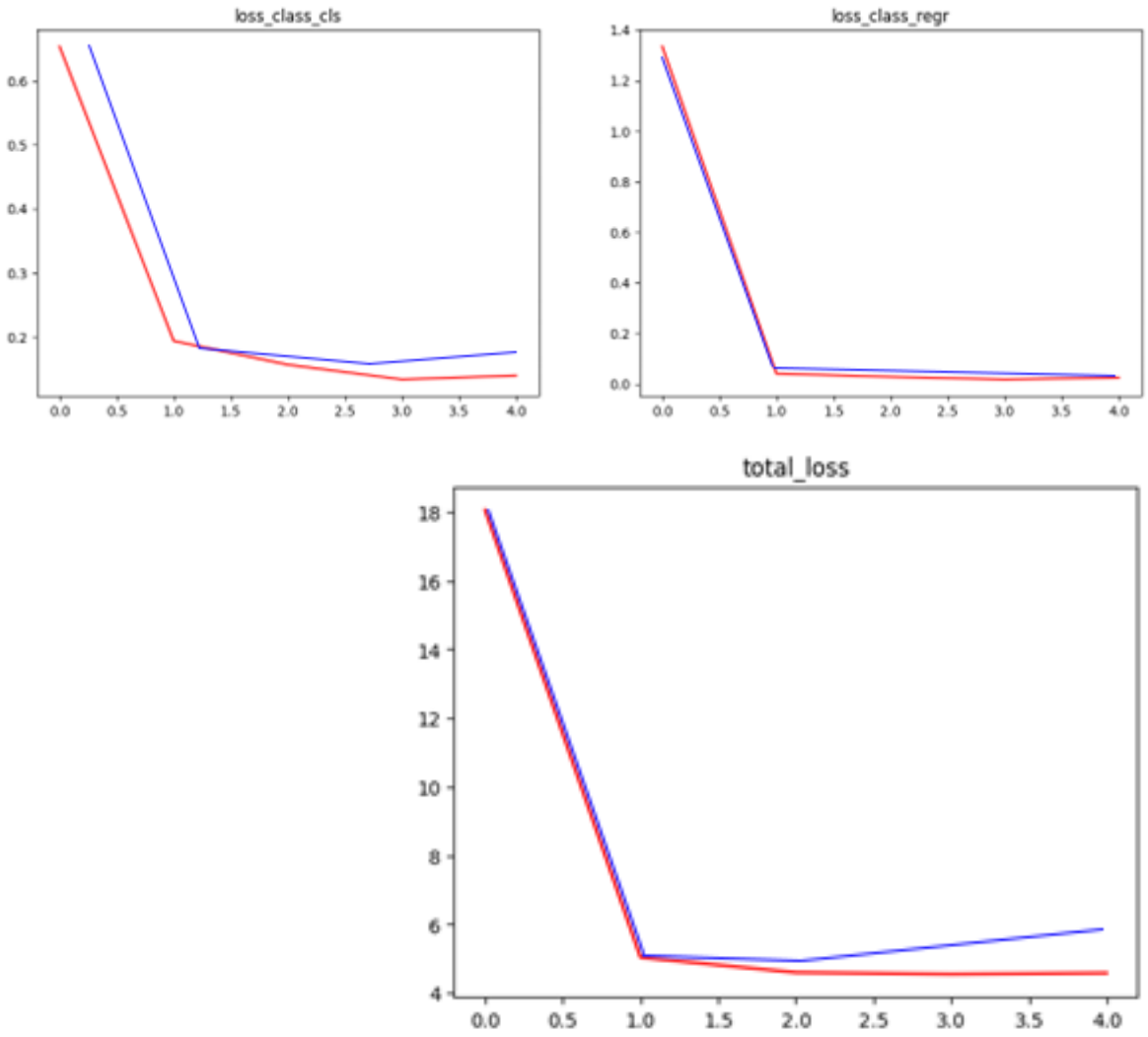
ROI Pooling:

این بخش از شبکه را نیز پیاده‌سازی کردیم.

۳-۲. آموزش شبکه

تمام مراحل گفته شده را انجام می‌دهیم و برای هر بخش از شبکه loss مربوط به آن را تعریف می‌کنیم، سپس مدل را برای ۵ دوره بر روی داده‌های آموزش، آموزش می‌دهیم. نتایج آموزش به شکل زیر خواهد شد:





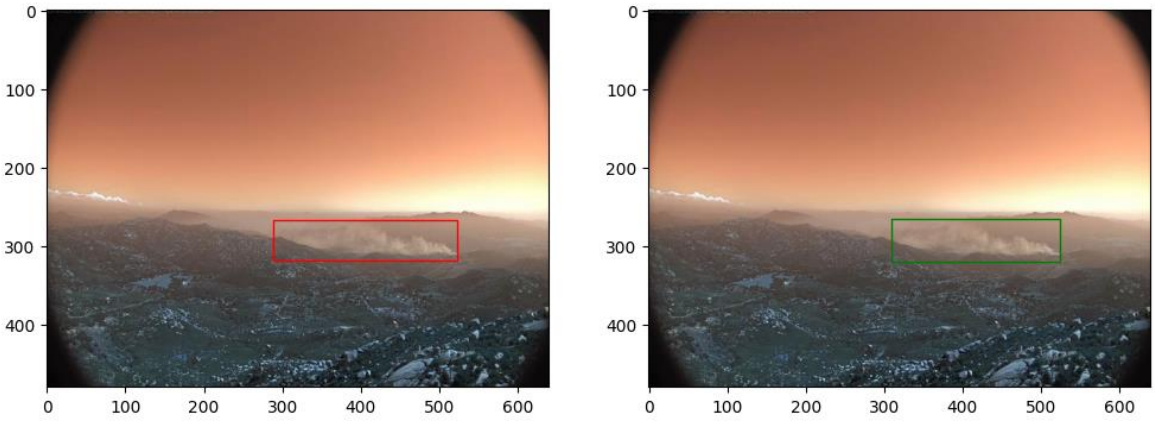
شکل ۱۶ نمودار خطای مدل

همچنین دقت و خطای مدل برای داده‌های تست به صورت زیر است:

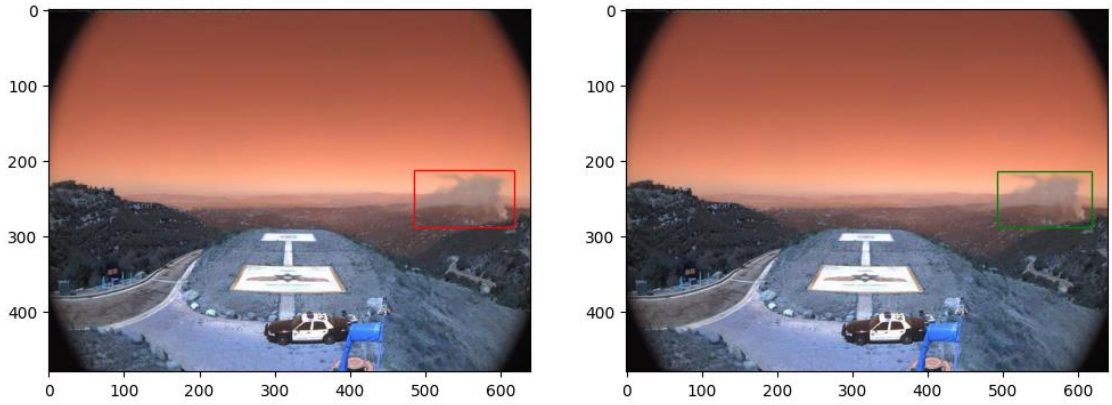
```
Loss RPN classifier: 4.018688185785897
Loss RPN regression: 0.3910871051396243
Loss Detector classifier: 0.15718835251964627
Loss Detector regression: 0.02840222899708897
Total loss: 4.595365872442256
```

در نهایت، کد با استفاده از مجموعه داده‌های آزمایشی، مدل را ارزیابی کرده و گزارش دقت، بازیابی و میانگین IoU را چاپ می‌کند. این اطلاعات به تحلیل‌گران کمک می‌کند تا عملکرد مدل را در شرایط واقعی بررسی و ارزیابی کنند.

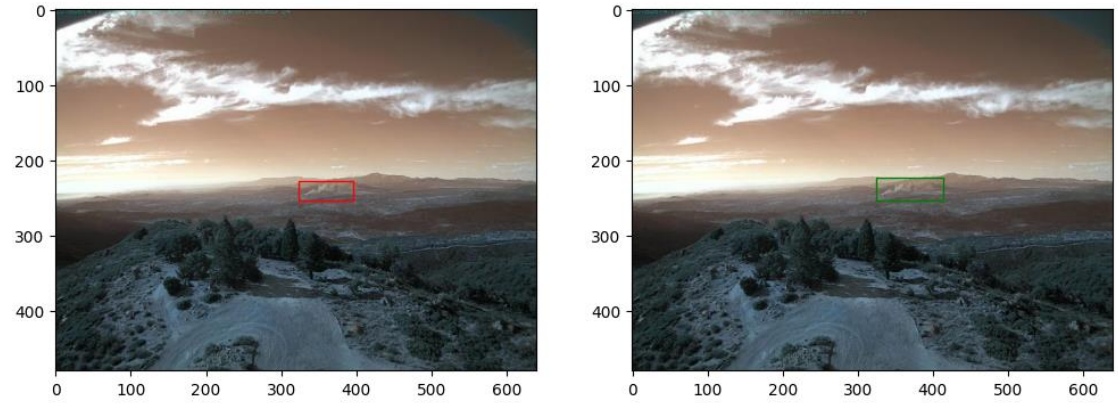
حال مثال ها به صورت زیر به دست می آید:



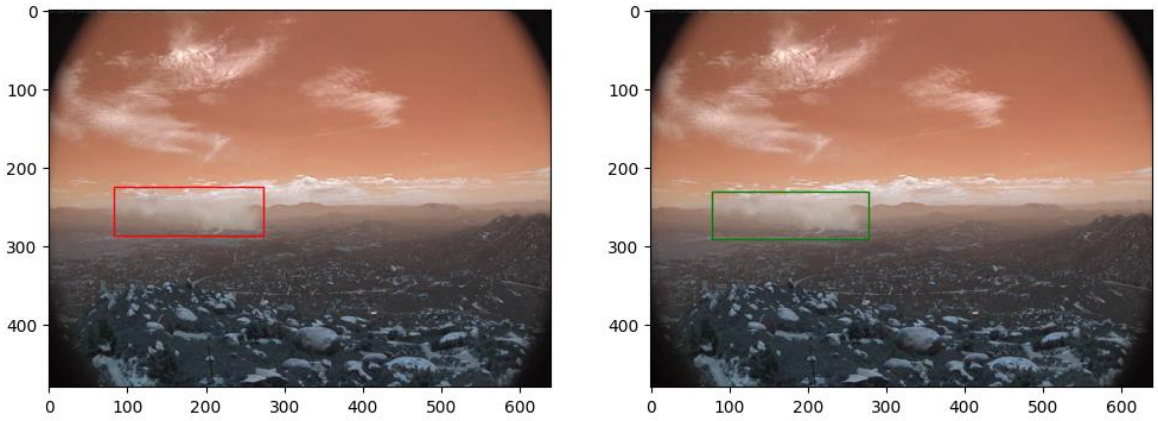
شکل **۱۷** : مثال اول



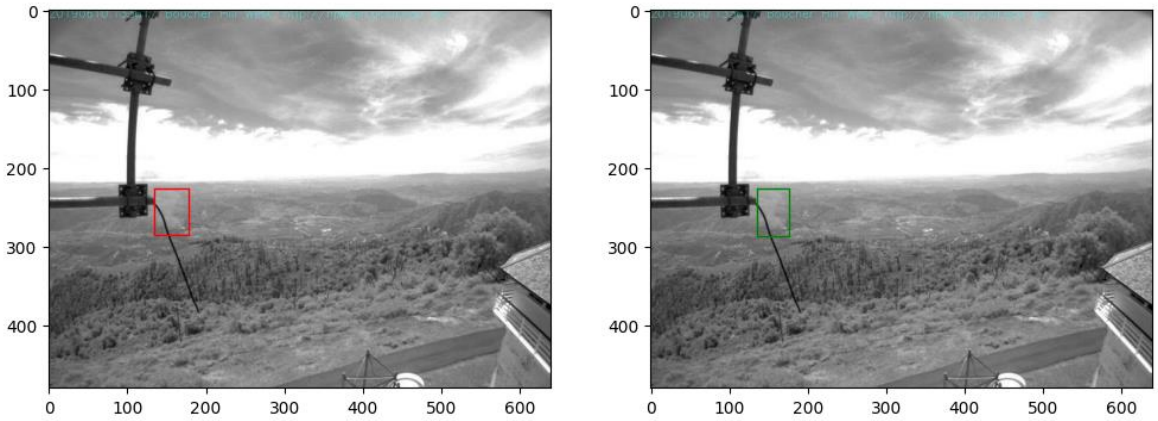
شکل **۱۸**: مثال دوم



شکل **۱۹**: مثال سوم



شکل **۲۰**: مثال چهارم



شکل **۲۱**: مثال پنجم

با توجه به نتایج مدل مشخص است که این مدل دارای دقت بالایی می‌باشد، اما LOSS این مدل نیز بالا می‌باشد پس ممکن است در داده‌های جدیدی که توزیع آن با توزیع این مجموعه‌داده متفاوت است به مشکل بخوریم. بنابراین برای بهبود عملکرد مدل این راه‌ها پیشنهاد می‌شود.

- بهتر کردن هدف optimization یا loss فانکشن‌ها
- روش‌های augmentation برای افزایش داده‌ها استفاده کنیم تا مدل دارای تعمیم پذیری بیشتری باشد.
- همچنین از backbone قوی تری نسبت به vgg مانند Resnet می‌توانیم استفاده کنیم. استفاده از ویژگی‌های بیشتر و افزایش توان محاسباتی مدل.