



## درس شبکه‌های عصبی و یادگیری عمیق

### تمرین سوم

e

نام نام خانوادگی - نام نام خانوادگی	نام و نام خانوادگی
810199357 - 810199395	شماره دانشجویی
1402.09.04	تاریخ ارسال گزارش

## فهرست

<b>1</b>	<b>پاسخ . SAM.....</b>	<b>1</b>
1	آماده سازی مجموعه داده 1-1.....	1
1	بارگذاری داده 2-1.....	1
1	تقویت داده 3-1.....	1
1	متريک و تابع هزينه و بهينه سازی 4-1.....	1
<b>3</b>	<b>پاسخ 2 . RCNN.....</b>	<b>3</b>
3	توضيحات مدل ها 2-1.....	3

## شکل‌ها

[۱. یک حفت نمونه از داده‌های مجموع](#)

[SAM model architecture.2](#)

[SAM model.3](#)

[loading the SAM model.۴](#)

[sam model output masks.4](#)

[Augmentaiton .5](#)

[\(without augment\(left\) vs augmentation\(right.6](#)

[Augmented images with their mask .7](#)

[...Sam Model results before tuning\(left\) vs after tuning .8](#)

[Sam Model results .9](#)

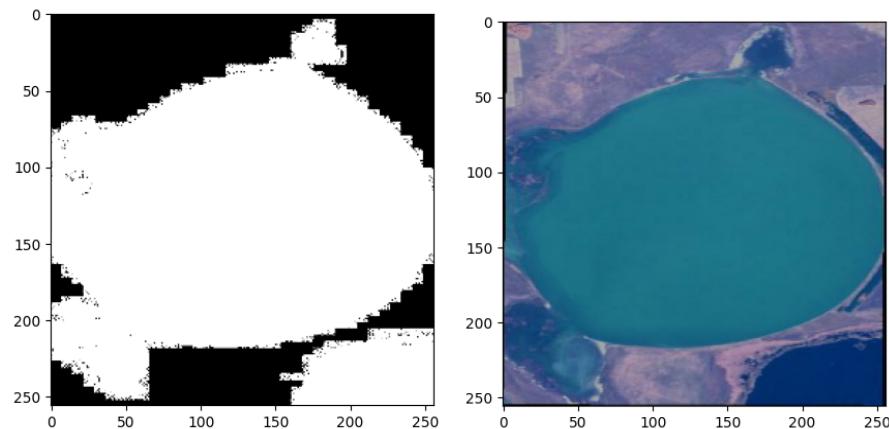
[FRCNN Preprocessing 10](#)

[Validation And Loss Diagaram .11](#)

## جدول‌ها



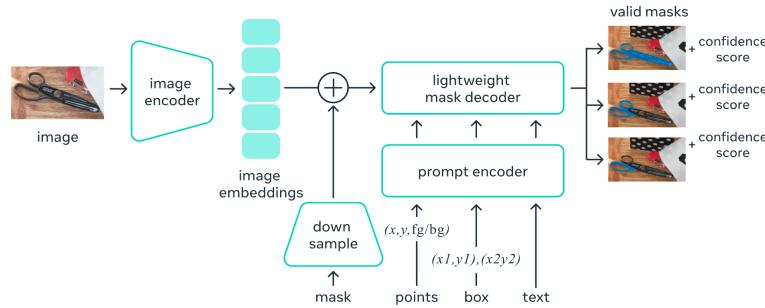
1-1. آماده سازی مجموعه داده



1. یک جفت نمونه از داده های مجموع

## ۱-۲. بارگذاری مدل

### Universal segmentation model



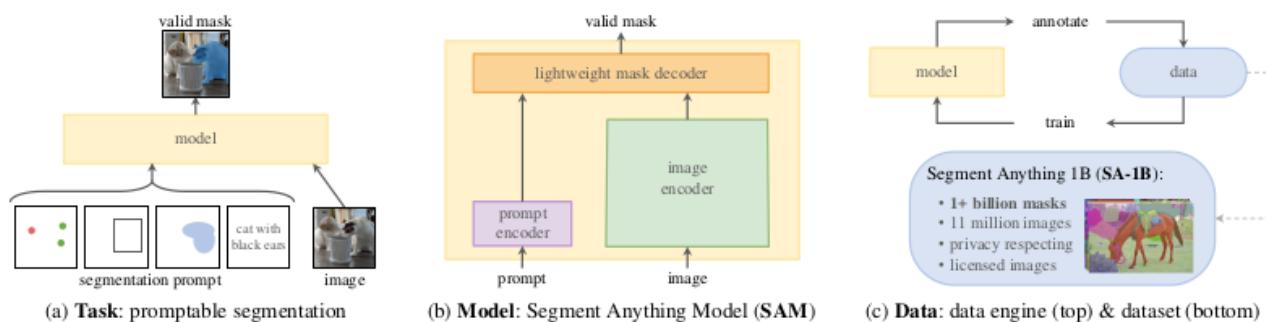
### 2.SAM model architecture

این مدل به روی ۱۱ میلیون عکس و ۱.۱ میلیارد mask آموزش داده شده است. این مدل عکس و یک prompt به عنوان ورودی دریافت میکند و مسک هایی برای تمام آبجکت هایی که می تواند آن ها را تشخیص دهد تولید میکند. این prompt میتواند مجموعه از نقاط یا bbox یا متن باشد. و اینکار ما مدل را به محدوده یا چیزی که میخواهیم راهنمایی میکنیم. البته هنوز نسخه prompt منن آن بیرون نیامده است. مدل به صورت semi-supervised آموزش دیده است. در واقع تعداد عکس به همراه target mask به مدل داده شده است و مدل به روی آن ها آموزش دیده است. سپس با کمک وضعیت فعلی مدل خروجی ها دیگری را تولید کردن و این خروجی ها توسط افرادی اصلاح شدند و دوباره به مدل feed شده اند و این حلقه تکرار شده است تا با کمک مدل بتوانند دیتا است بزرگی فراهم کنند و مدل را به روی آن آموزش دهند. در حال حاضر میتواند مدل به صورت zero-shot,few-shot در زمینه های مختلف استفاده کرد.

#### :image encoder بخش

این بخش که همان image encoder در عکس بالا میباشد، در واقع یک عکس را به عنوان ورودی میگیرد و آن را تبدیل میکند به یک  $256 \times 64 \times 64$  feature matrix. این فیچر ها به mask decoder داده میشوند. تا مدل براساس آن و prompt برای ما خروجی را تولید کند. یکی از فواید ساختار این مدل این هست که فاز سنگین image featurization backend model محول کرده است. و به روی آن یک مدل سیکتر را آموزش دیده است. با اینکار میتوان بخش اول روی یک gpu اجرا کرد و بخش سبکتر سمت client web browser اجرا شود.

### 3.SAM model



بخش ۳ image encoder نسخه دارد، vit-h و vit-b. که به ترتیب تعداد پارامتر ها بیشتر میشود. مقدار پارامتر های هر یک برابر با: vision transformer-base/large/huge

(ViT-B (91M), ViT-L (308M), ViT-H (636M parameters)  
ما در اینجا از vit-base استفاده میکنیم.

```
from transformers import SamModel, SamProcessor
sam = SamModel.from_pretrained("facebook/sam-vit-base")
processor = SamProcessor.from_pretrained("facebook/sam-vit-base")
img_url = "https://huggingface.co/ybelkada/segment-anything/resolve/main/assets/car.png"
raw_image = Image.open(requests.get(img_url, stream=True).raw).convert("RGB")
# raw_image = Image.open("/home/sepehr/Downloads/car.png").convert("RGB")
raw_image
```

config.json: [6.57k/6.57k [00:00<00:00,  
100% 571kB/s]

pytorch\_model.bin: [375M/375M [00:01<00:00,  
100% 309MB/s]

preprocessor\_config.json: [466/466 [00:00<00:00,  
100% 42.3kB/s]

[6]:



۱.loading the SAM model



#### 4.SAM model output masks

### 3-1. تقویت داده

استفاده از لایبریری augmentations برای کردن داده ها.

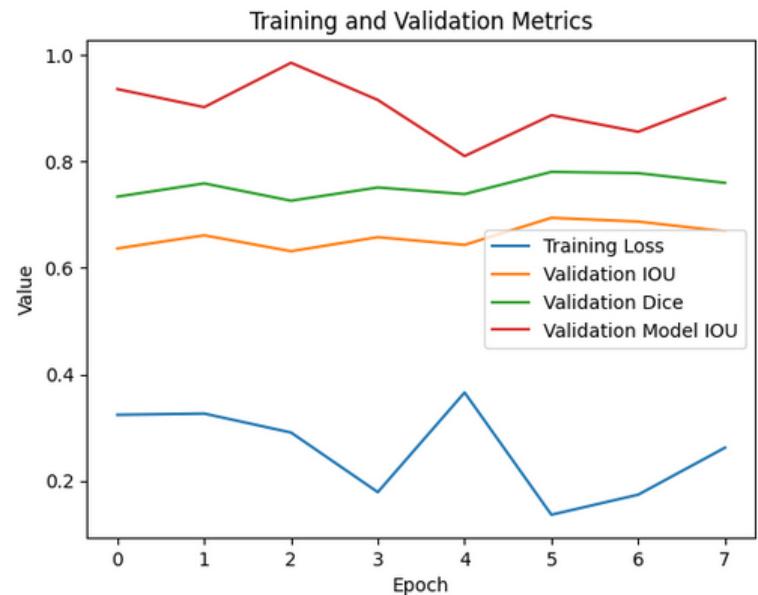
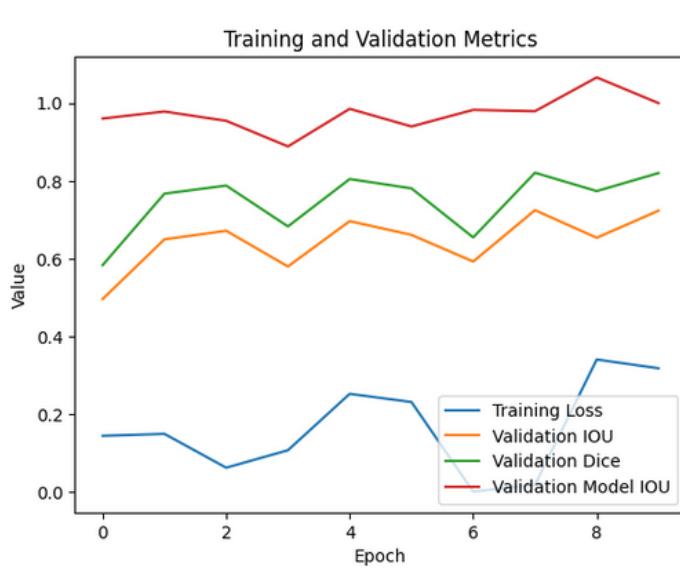
```

augmentations = [
    A.HorizontalFlip(p=1),
    A.VerticalFlip(p=1),
    A.Affine(rotate=90, p=1.0),
    A.Affine(rotate=180, p=1.0),
    A.Affine(rotate=270, p=1.0),
]

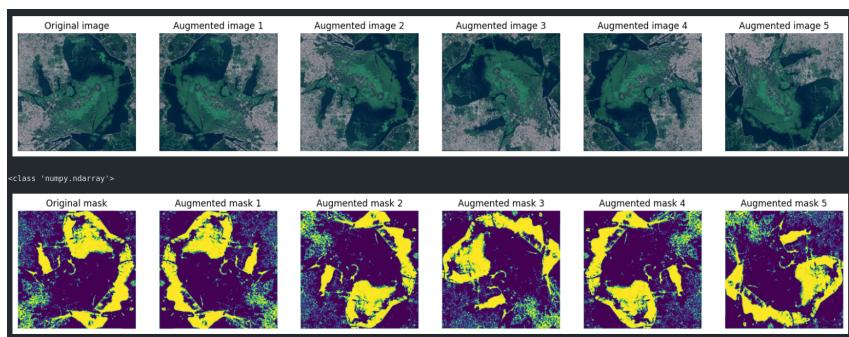
```

## 5. Augmentation

در اینجا از ۵ روش برای تولید داده استفاده شده که در واقع چرخش ۹۰ درجه و فلیپ کردن عکس هست. از آنجایی که عکس ها جهتی ندارند. در نتیجه عکس های تولید شده، عکس های معقولی برای feed کردن به مدل میباشند. با اینکار مدل generalization بیشتری بدست میاورد. (در حالت اگمنت به علت نبود ریسورس، نتوانستیم مدل را به تعداد کافی ران کنیم. اما همچنان به خوبی حالت اگمنت نشده، مدل آموزش دیده است)

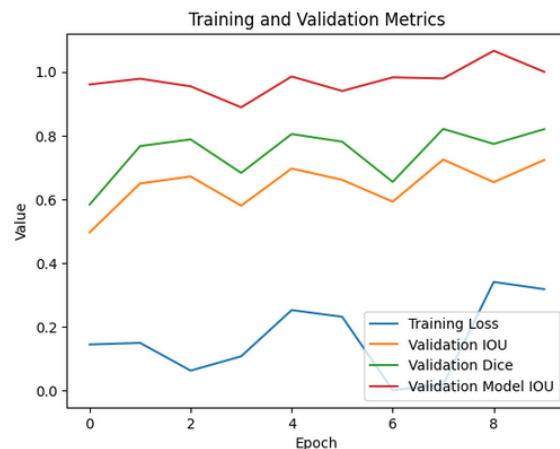


6. without augment(left) vs augmentation(right)

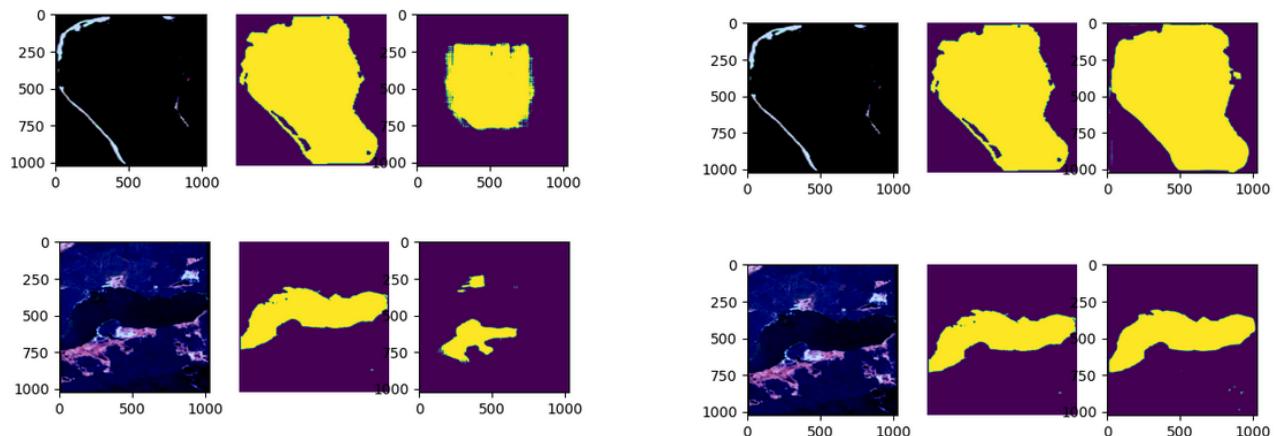


7. Augmented images with their mask

#### 4-1. بهینه ساز ، متریک وتابع هزینه



اولی mask ground truth است و mask predication مدل است. که بعد از tuning بهتر شده است.



7. Sam Model results before tuning(left) vs after tuning(right)

```
-----epoch6-----
Epoch 7: 100%|██████████| 640/640 [13:23<00:00,  1.26s/it]
end of epoch
Number of parameters changed: 96
Epoch 6, Loss: 0.000446885242034912
Average IoU: 0.5923441052436829, Average Dice Coefficient: 0.6544040441513062,Average Model IoU: 0.9825071096420288
-----epoch7-----
Epoch 8: 100%|██████████| 640/640 [13:23<00:00,  1.26s/it]
end of epoch
Number of parameters changed: 96
Epoch 7, Loss: 0.0180734097957611
Average IoU: 0.7244924902915955, Average Dice Coefficient: 0.8207795023918152,Average Model IoU: 0.9790769219398499
-----epoch8-----
Epoch 9: 100%|██████████| 640/640 [13:21<00:00,  1.25s/it]
end of epoch
Number of parameters changed: 96
Epoch 8, Loss: 0.34032049775123596
Average IoU: 0.6535344123840332, Average Dice Coefficient: 0.7735952138900757,Average Model IoU: 1.066007375717163
-----epoch9-----
Epoch 10: 100%|██████████| 640/640 [13:20<00:00,  1.25s/it]
end of epoch
Number of parameters changed: 96
Epoch 9, Loss: 0.31772640347480774
Average IoU: 0.7232049703598022, Average Dice Coefficient: 0.8199214935302734,Average Model IoU: 0.9998964667320251
```

#### 8. Sam Model results

### 1-2. توضیحات مدل ها

#### پاسخ 2-1 . توضیحات مدل ها

##### Region Based CNNs •

این گونه از CNN ها با استفاده از Region Proposal هایی پیدا کنند که در داخل شاین Bounding Box ها سعی می کنند که در داخل شاین Feature Extraction باشند . سپس معماری کلی آن ها به این شکل است که این feature های استخراج شده را از چند Fully Connected Layer رد می کنند تا برای مثال Classification برای Object Detection انجام دهند .

مزایا : یک Breakthrough در زمینه Object Segmentation حساب می شوند

##### معایب

۱. حافظه بسیار زیادی نیاز دارد

۲. سرعت آن ها به دلیل Redundant Calculation بسیار کم است (هر Region باید یک بار به داخل Neural Network Feed شود )

۳. فرآیند Train کردن دشوار و پیچیده است چون از مراحل متفاوتی همچون Box Regressor و Classifier و خود CNN تشکیل شده است .

نکته درباره این نوع از RCNN ها این است که مدل یکپارچه ای ندارند و مرحله Preprocessing Region Proposal یک نوع Region Proposal است که با الگوریتم های یادگیری ماشین غیر از آموزش می بینند

##### Fast Region Based CNNs •

یک CNN برای تولید Feature Map استفاده می کند تا تعداد Redundant Computation ها را کاهش دهد . این معماری از یک مازول به نام RPN استفاده می کند که feature map های یک شبکه را همراه با خود عکس ها و یک مجموعه از Anchor ها گرفته و یک سری Region های پیشنهادی را پیشنهاد می کند این ناحیه ها می توانند برای استخراج Bounding Box استفاده شوند .

Anchor Generator یک مازول است که در واقع به ازای هر پیکسل چندین Anchor تولید کرده که برای تولید ناحیه ها استفاده می شود . نیاز به Anchor برای این است که میخواهیم تشخیص دهیم را با Ratio به مدل ایده ای بدھیم که دنبال چی باشد .

##### مزایا نسبت به RCNN

پروses آموزش آسان تر و یکپارچه است .

تر است چون تعداد کمتری Redundant Region تولید می کند و از یک ناحیه برای تشخیص چندین شی می تواند استفاده کند و آموزش میبینند که ناحیه هایی را پیشنهاد دهد که بیشتر اشیا در آن ها هستند و همچنین برای ناحیه های شلوغ می تواند Redundant Computation های بزرگتری که شامل اشیا بیشتری هستند پیشنهاد بدهد تا از جلوگیری شود .

##### Faster Region Based CNNs •

این گونه از RCNN ها مانند قبل عمل می کنند با اضافه کردن یک مازول ROI سعی می کنند که Redundancy را باز هم کاهش بدھند به این گونه که اول از یک لایه CNN داده را گذر می دهد تا feature map را تولید کند سپس با استفاده از لایه RPN سعی می کند یک سری region proposal تولید کند که تعداد کمتری feature extraction برای مدل نیاز باشد .

در این معماری از لایه object detection و bounding box detection هم برای feature extraction استفاده می شود و در نتیجه سریع تر است .

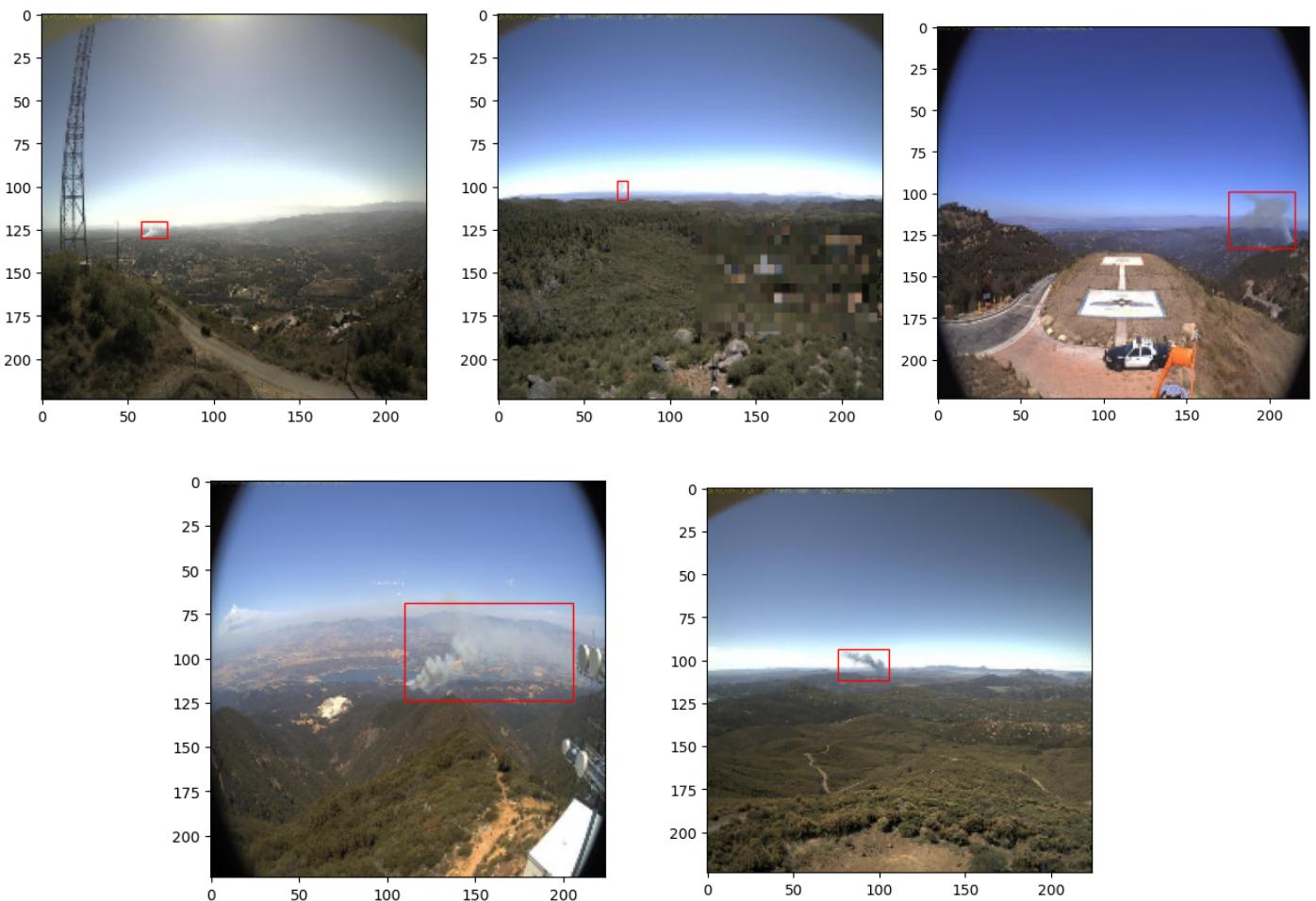
ماژول ROI یک نوع Max pooling رو ناحیه های بزرگتر پیشنهاد شده توسط RPN انجام می دهد و آن ها را به سایز های کوچک تر ( مثلا ۷ در ۷ ) می رساند که Bounding Box از داخل آن ها یا مجموعه ای از این عکس های کوچک تر قابل استخراج است.

**FRCNN**  
مزایا نسبت به  
پروسه آموزش سریع تر است .  
Computationally Light Weight تر است .

## کاربرد لایه ها

از RPN برای تشخیص دادن از قبل Region هایی استفاده می شود که قبل از Classification احتمال تشخیص Object در آن ها بالاتر است . این لایه یک نوع ورودی برای ROI Pooling استفاده می کند از ROI های پیشنهادی در هر لایه استفاده کرده تا بلوک های سایز ثابتی برای لایه های بعد تولید کند . ROI Pooling در هر منطقه تولید شده توسط لایه های قبل کار Classify کردن آن لایه رو به عهده دارد .

## پاسخ ۲-۲ . Preprocessing



10. 5 نمونه از تصاویر دیتابست قابل مشاهده هستند که به ۲۲۴x۲۲۴ rescale شده اند همچنین Bounding Box مربوط با این عکس ها نیز Scale شده اند . در کل همه عکس های دادگان شامل یک دود حاصل از آتش گرفتن جنگل ها است و ما دنبال Bounding Box در نهایت هستیم .

## پاسخ ۲-۳ . آموزش دادن مدل

در نمودار و اشکال زیر Train شدن FRCNN را مشاهده می کنیم

پارامتر هایی که در مقاله داده شده بودند :

که همین مقدار سنت شده که تو مقاله سنت شده بود.

Sampling Ratio ۲ سنت شده که البته می تواند بالاتر برود ولی مدل کنترل ترین می شود  
output size = 7 سنت شده است. که در واقع سایز خروجی های ROI خواهد شد .

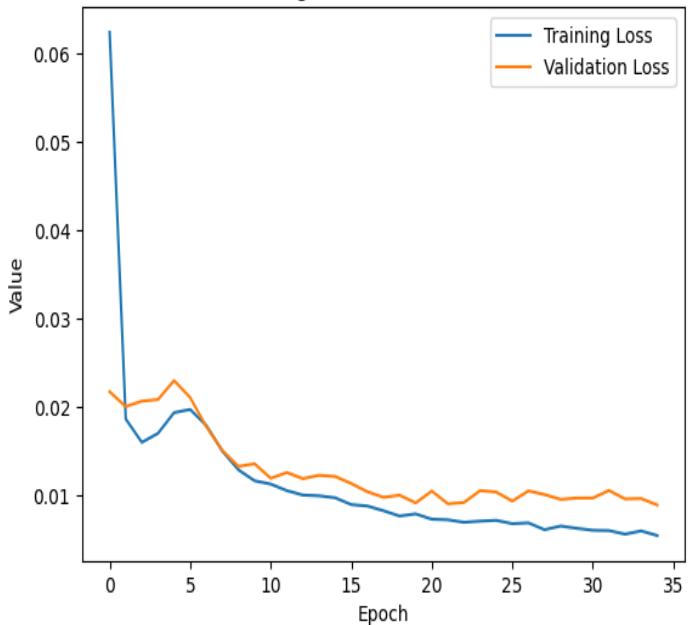
ها در مقاله داده نشده بودند که ما تعدادی Anchor در نظر گرفته و سنت کرده ایم در نهایت توضیح میدهیم که برای دادگان : ( Data Analysis challenging است و کار بیشتری نیاز دارد ( کار smoke aspect ratio

```
anchor_generator = AnchorGenerator(  
    sizes=((32, 64, 128, 256, 512),), aspect_ratios=((0.5, 1.0, 2.0),))
```

### Training & Validation Loss .11

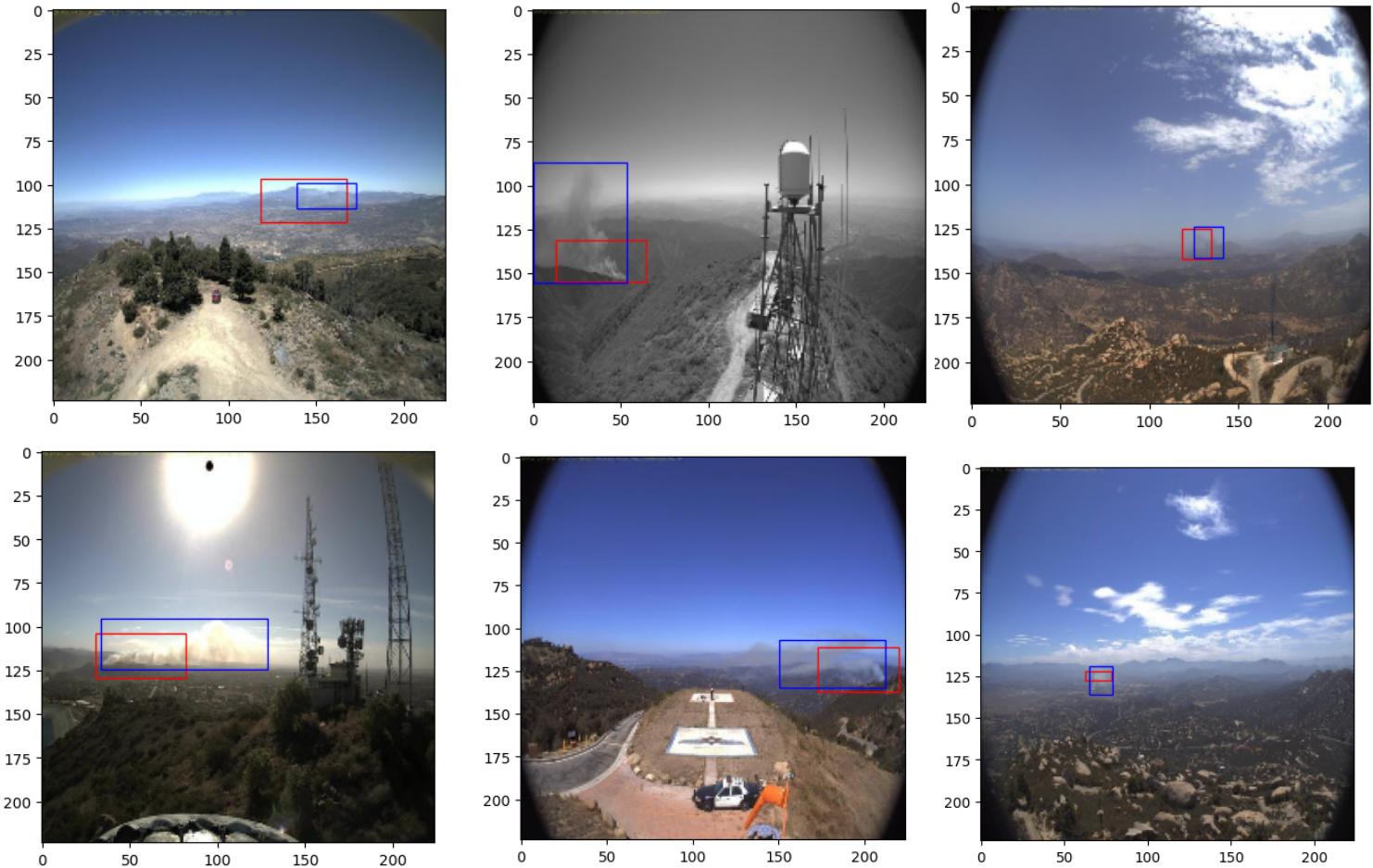
```
-----epoch0-----  
Epoch 1: 100%|██████████| 129/129 [00:55<00:00, 2.33it/s]  
  
Epoch [1/35], Train_Loss: 0.062362122594905915  
  
Epoch [1/35], Val_Loss: 0.021734264238100303  
-----epoch1-----  
Epoch 2: 100%|██████████| 129/129 [00:50<00:00, 2.55it/s]  
  
Epoch [2/35], Train_Loss: 0.018648092548341252  
  
Epoch [2/35], Val_Loss: 0.020056824052804394  
-----epoch2-----  
Epoch 3: 100%|██████████| 129/129 [00:58<00:00, 2.53it/s]  
  
Epoch [3/35], Train_Loss: 0.016009820964505043  
  
Epoch [3/35], Val_Loss: 0.020664626389349763  
-----epoch3-----  
Epoch 4: 100%|██████████| 129/129 [00:50<00:00, 2.53it/s]  
  
Epoch [4/35], Train_Loss: 0.017027464059954003  
  
Epoch [4/35], Val_Loss: 0.020848647691309452  
-----epoch4-----
```

### Training and Validation Metrics



## پاسخ ۲-۴ . عملکرد روی دادگان تست

در زیر برخی از خروجی ها روی دادگان تست را مشاهده می کنیم که در عکس های زیر bounding box واقعی آبی و bounding box پیش بینی شده قرمز است.



یکی از چالش های اصلی تشخیص دادن object ها این است که مدل بفهمد که دنبال object هایی از چه aspect ratio هستند . دلیل اینکه در Anchor خود aspect ratio استفاده می کنیم همین است .

یکی از چالش های اصلی همین سوال نیز این است که دوD aspect ratio ثابتی ندارد و expand می کند در واقع مدل خیلی جاها توانسته است محل تخمینی دوD را تخمین بزند ولی نتوانسته طول درستی از این دوD ها را در بیشتر case ها تشخیص دهد .

احتمالا با pre train یک مدل برای تعیین کردن Anchor های بهتر در مرحله Hyper Parameter Tuning بتوانیم نتیجه بهتری با تعیین کردن Aspect Ratio و طول های بهتر داشته باشیم.