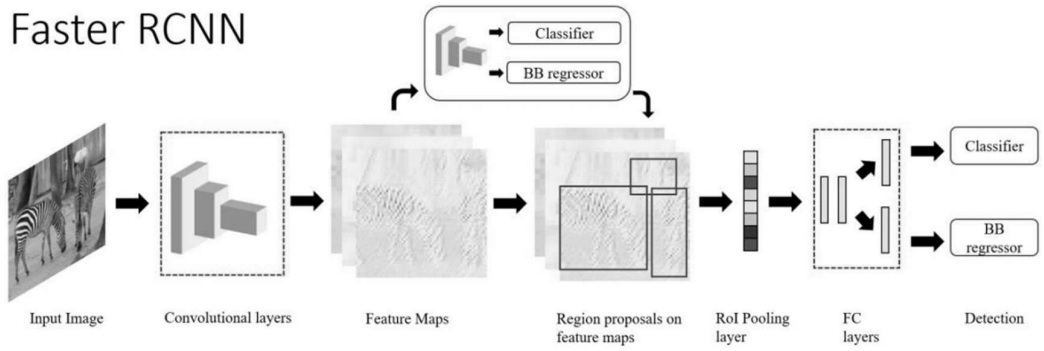
پاسخ ۲ – تشخیص و شمارش اشیا

۲-۱. معماری شبکه

معماری Faster R-CNN از دو بخش اصلی تشکیل شده است: یک شبکه Region Proposal Network (RPN) و یک تشخیص‌دهنده Fast R-CNN. این معماری برای شناسایی شیء در دو مرحله طراحی شده است: ابتدا تولید مجموعه‌ای از پیشنهادات نواحی که به احتمال زیاد شامل اشیاء هستند، و سپس طبقه‌بندی و بهبود پیشنهادات شیء.

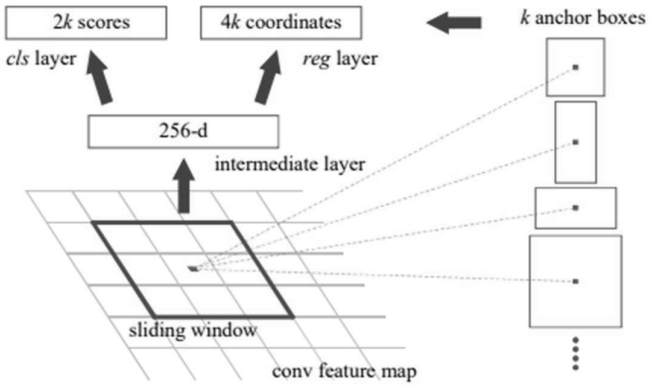


شکل ۱۵: شمای شبکه **Faster-RCNN**

از این معماری به شرح زیر استفاده می‌شود:

۱. لایه‌های کانولوشنی: تصویر ورودی از طریق چندین لایه کانولوشنی عبور می‌کند این لایه ها برای استخراج فیچرهای تصویر کاربرد دارد

۲. شبکه RPN: در این معماری از یک شبکه پیشنهاد ناحیه دارای شیء(Region Proposal Network) استفاده شده و در آن با استفاده از Anchor box (که مرزهای از پیش تعریف شده هستند) به احتمال حضور اشیاء در کلیه نواحی، امتیازی اطلاق میکند. نواحی دارای امتیاز بالا به عنوان اشیاء پیشنهادی انتخاب می‌شوند. ضرایب این قسمت باید آموزش داده شوند.



شکل ۱۶: حرکت **sliding window** بر روی فیچر مپ‌های تصویر

۳. RoI-pooling: بر نواحی پیشنهادی اشیاء تجمیع RoI اعمال میشود تا یک نقشه ویژگی با اندازه ثابت از هر ناحیه پیشنهادی استخراج شود.

۴. لایه دنس تشخیص‌دهنده :پس از تولید پیشنهادات شیء، تصویر هر پیشنهاد از طریق یک شبکه عصبی کانولوشنی پردازش می‌شود تا ویژگی‌های مربوط به شیء استخراج شوند. سپس این ویژگی‌ها به لایه‌های fully connected می‌روند و برای یک کلاس‌بندی و یابنده‌ی Bonding Box عمل می‌کنند. در نهایت، هر پیشنهاد با یک برچسب و مختصات جعبه متناظر با آن برچسب گذاری می‌شود.

۲-۲. پیاده سازی شبکه

برای دریافت مدل پس از استفاده تابع گفته شده در صورت سوال لازم بود تا معماری لایه‌های آخر بر اساس تعداد کلاس ها تغییر کند. تابعی به این منظور نوشته شد.

```
def get_model(num_classes):
    # Load the pre-trained Faster R-CNN model and modify the output layer
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    model.roi_heads.box_predictor = torchvision.models.detection.faster_rcnn.FastRCNNPredictor(in_features, num_classes)
    return model
```

شکل ۱۷. تابع دریافت مدل و تغییر لایه آخر

از تابع ترنسفورم داده شده استفاده شد تا تصاویر به تنسور تبدیل شوند.

همچنین از کلاس pascal-dataset برای وارد کردن داده استفاده شد.

```
# Define the training and testing datasets
train_dataset = PASCALDataset('/content/drive/MyDrive/HW3/PASCAL/train')
test_dataset = PASCALDataset('/content/drive/MyDrive/HW3/PASCAL/val')
```

شکل ۱۸: دریافت داده های ترین و ولیدیشن

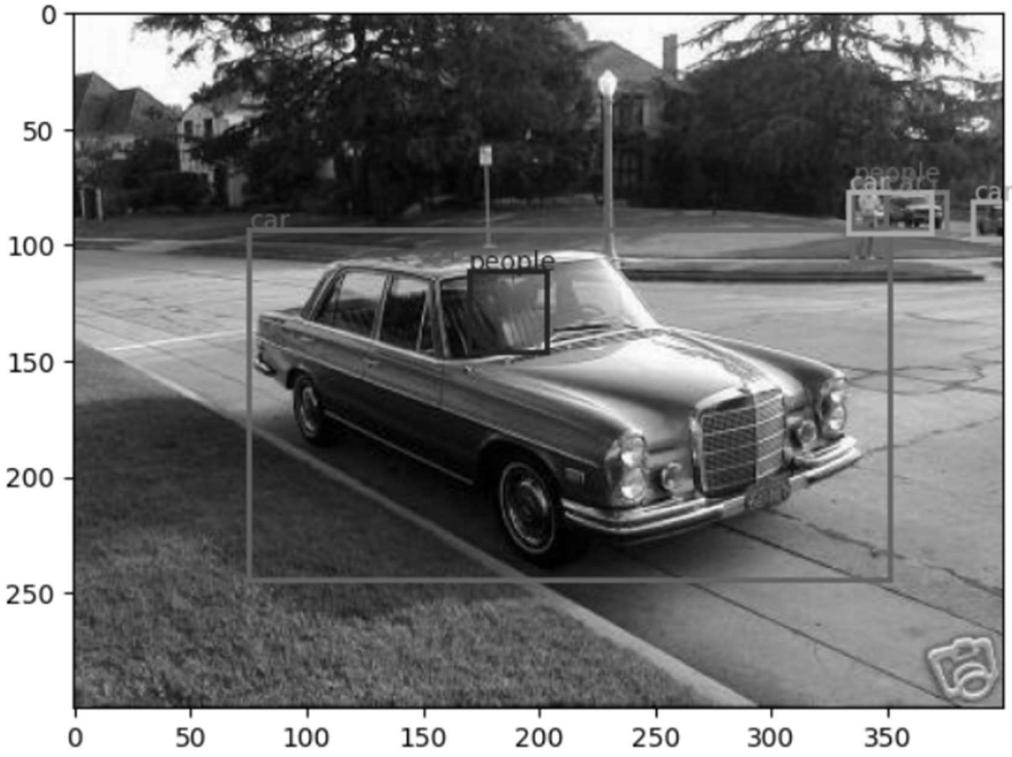
تعداد کلاسها همراه با کلاس پسزمینه ۶ تا در نظر گرفته شد. و مدل تعریف شده به gpu ارجاع شد. سپس مدل برای ۵ epoch آموزش داده شد و از تابع evaluate در فایل engine قرار داشت کیفیت آموزش بررسی شد.

```
Test: Total time: 0:00:24 (0.1231 s / it)
Averaged stats: model_time: 0.1090 (0.1115)  evaluator_time: 0.0016 (0.0020)
Accumulating evaluation results...
DONE (t=0.08s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.516
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.859
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.567
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.227
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.464
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.605
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.428
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.622
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.625
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.320
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.606
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.686
```

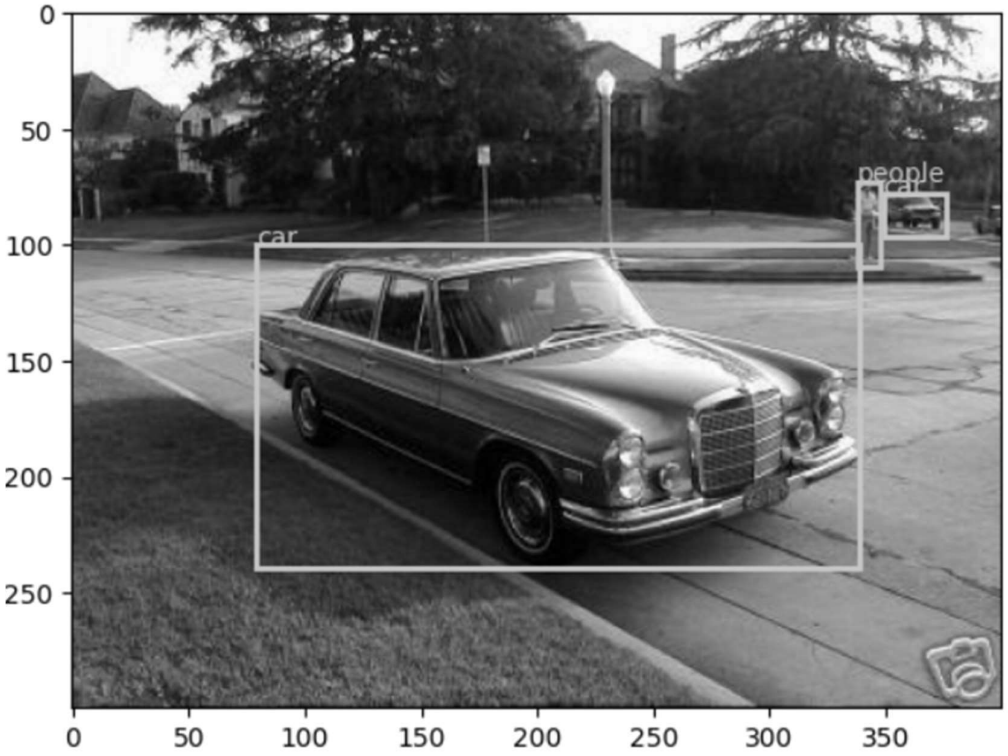
شکل ۱۹: عملکرد آموزش با تابع **evaluate**

۲-۳. تست شبکه

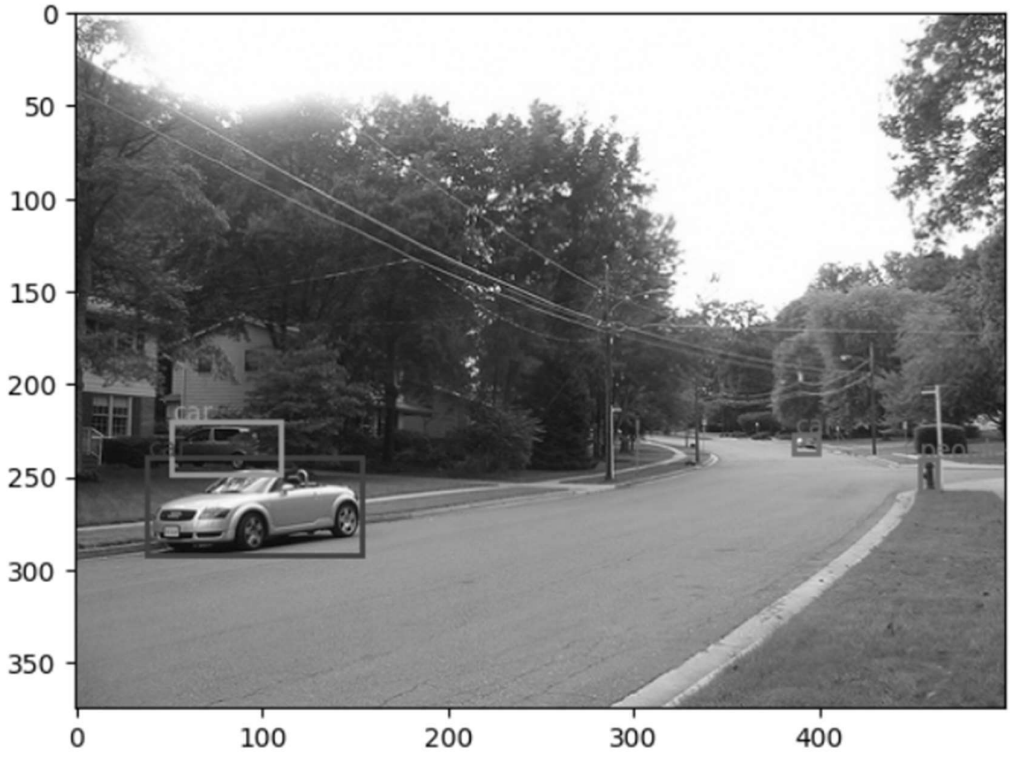
تصاویر زیر عملکرد مدل برای استخراج اشیاع و نامگذاری آن ها بعد از آموزش را با تصاویر لیبل گذاری شده‌ی داده های تست نشان میدهد. عملکرد شبکه با وجود خطاهای منطقی مطلوب به نظر میرسد. تصاویر بالایی همان MODEL OUTPUT آموزش داده شده و تصاویر پایین همان EXPEXTED OUTPUT هستند.



شکل ۲۰: عملکرد مدل بر یکی از تصاویر دادگان تست

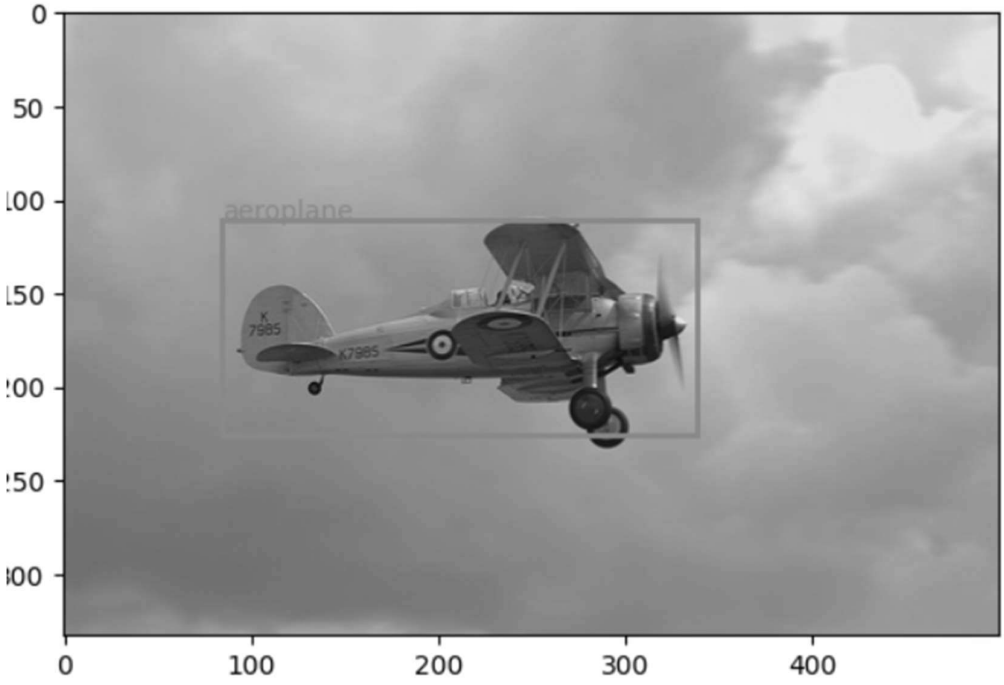


شکل ۲۱. عملکرد مورد انتظار بر یکی از تصاویر دادگان تست



شکل ۲۲. عملکرد مدل بر یکی از تصاویر دادگان تست

شکل ۲۳: عملکرد مورد انتظار بر یکی از تصاویر دادگان تست



شکل ۲۴. عملکرد مدل بر یکی از تصاویر دادگان تست

شکل ۲۵. عملکرد مورد انتظار بر یکی از تصاویر دادگان تست