# Target Detection

## – Didi Chuxing: Object Detection and Recognition in Traffic Scene


## Abstract

The task is to use deep learning technology to complete object detection and recognition in the city traffic scenes. The model should be able to find out where targets are located in the area and which group they are associated with, such as van, car, bicycle and so on. Dataset used in this project is a collection of images generated from a D2-City large-scale driving record video dataset, through a series of pre-processing steps, including frame extraction, etc.

Due to the limitation of computation strength and time budget, not many network architectures or components for Faster R-CNN are tested. After some simple parameter tuning, we tried other networks such as Cascade R-CNN, Dynamic R-CNN and Faster R-CNN with FCOS as head. In addition, we switched the network(backbone) and other components to investigate how they can affect the learning results.

With anchor scale set to [4, 8] and using IoU Loss as the bounding box loss, Faster R-CNN has the best Average Precision and Average Recall scores among the models we tested, under a limited epoch amount. When IoU is 0.5, the AP is 0.479 and when IoU is 0.75, the AP is 0.301. For IoU in the range of 0.5 to 0.95, the AP reaches 0.291. This model has the highest AP 0.325 for medium size object detection. For small and large size objects, the APs are 0.130 and 0.450. Although most of its AP scores are not the highest, it is still one of the best, together with high AR scores acrossing all IoU ranges.

If time applies, it would be worth a try to see how non-RCNN family models behave on our dataset. It would also be interesting to dig in deeper to all networks we tested, especially when changing the backbone/head/neck for Faster R-CNN. The Cascade R-CNN and Dynamic R-CNN have comparatively weaker performances compared to how some materials suggest. We should run more tests and tunings to validate and analyze this phenomenon.


## 1 Introduction

Object detection is the process of finding instances of objects in images. In the case of deep learning, object detection is a subset of object recognition, where the object is not only identified but also located in an image[1]. It is widely used in computer vision tasks such as image annotation, vehicle counting, face detection, face recognition. It is also used in tracking objects. In our case, we want the detectors to recognize objects under traffic scenarios and where they are as accurately and completely as possible. Common algorithms for such tasks are: Fast R-CNN, Faster R-CNN, YOLO, R-FCN etc[2].

Our dataset contains 10000 images for the train set, 1000 for validation and 1000 for test and inference. The train and validation set provide both the original image data and label information, whilst the test set only contains images. Images are all from a video dataset named ‘D2-City’, a large-scale dashcam video dataset of diverse traffic scenarios[3]. For ease of use, the labeled information of the dataset has been preprocessed into MS-COCO format[4]. The dataset contains a total of 12 object categories to be detected.

The learning results are evaluated by mAP(mean average precision). It computes the average precision value for recall value over 0 to 1. For COCO, AP is the average over multiple IoU (the minimum IoU to consider a positive match). AP@[.5:.95] corresponds to the average AP for IoU from 0.5 to 0.95 with a step size of 0.05. For the COCO competition, AP is the average over 10 IoU levels on 80 categories (AP@[.50:.05:.95]: start from 0.5 to 0.95 with a step size of 0.05)[5]. Below is an example for AP results for the YOLOv3 detector. Basically, in object detection tasks, precision is calculated based on the IoU threshold.

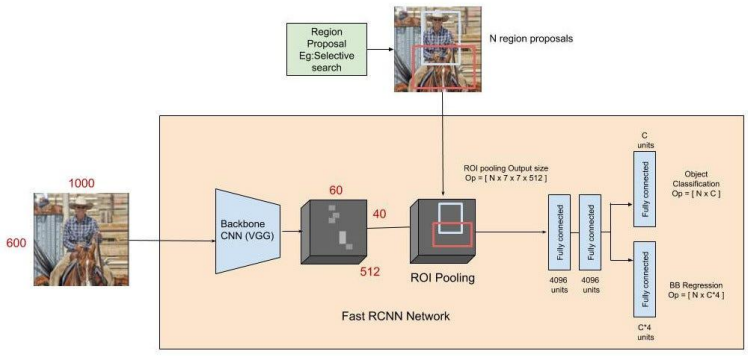
| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [3] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [6] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [4] | Inception-ResNet-v2 [19] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [18] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [13] | DarkNet-19 [13] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [9, 2] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [2] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [7] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [7] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

COCO for YOLOv3

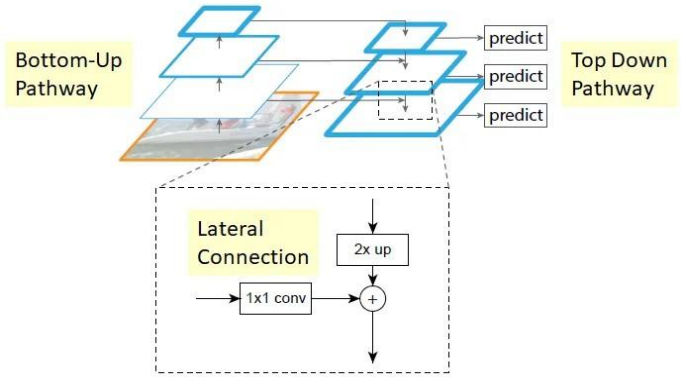
## 2 Programming Modules

MMDetection is an open source object detection toolbox based on PyTorch. It consists of training recipes for object detection and instance segmentation. Pre-trained models and popular dataset support[6]. It uses a modular design, all modules with different functions can be configured through the config. Visualization for training data is generated by the analysis tools built in the MMDetection.

## 3 Faster R-CNN




Left is an architecture diagram for Faster R-CNN with RPN(Region Proposal Network). Among the series of R-CNN models, the evolution is usually in terms of computational efficiency, reduction in test time and improvement in performances (mAP). These networks usually




consist of — a) A region proposal algorithm to generate “bounding boxes” or locations of possible objects in the image; b) A feature generation stage to obtain features of these objects, usually using a CNN; c) A classification layer to predict which class this object belongs to; and d) A regression layer to make the coordinates of the object bounding box more precise[7]. Compared to R-CNN and Fast R-CNN, two CPU-based region proposal algorithms, Faster R-CNN fixes this limitation by introducing in another convolutional network(RPN) to generate the region proposals.

In our original code framework, we use Feature Pyramid Network (FPN) rather than RPN. Unlike another “pyramid-like” architecture, the Pyramidal Feature Hierarchy, FPN enables the opportunity to reuse the higher-resolution maps of the feature hierarchy. This benefits the detection of small objects. The left diagram describes the architecture of a FPN network. It combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections[8].

## 4 Faster R-CNN with ResNet50 and FPN

Besides the general parameter tuning on max epoch and batch size, this project will mainly focus on how the change of Faster R-CNN components can impact the learning results and training process, for example, backbone, bottleneck, loss functions and optimizers. We will also test different sizes of anchor boxes for sliding through a given image.

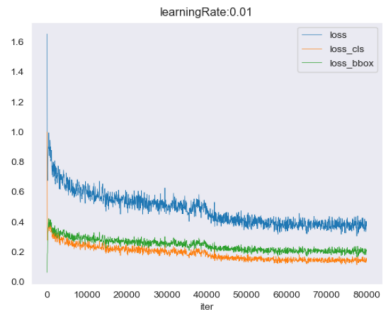
The base settings for testing are: SGD optimizer with momentum 0.9 and weight decay 0.0001, Cross Entropy loss and L1 Loss for bounding box. The L1 loss measures the mean absolute error (MAE) between each element in the two input matrices. The initial batch size is 2 and the max epoch is 16.

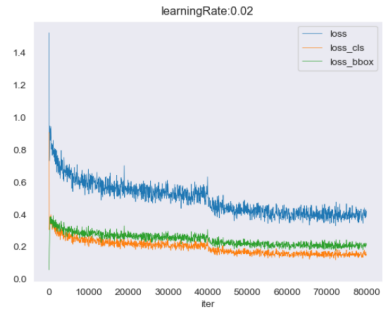
### 4.1 Learning Rate

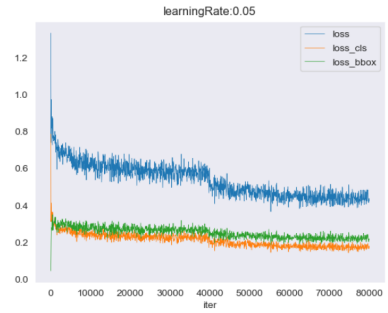
In this section, we will use SGD optimizer with momentum and weight decay. The loss function is CrossEntropy, and the type of bounding box loss is L1 Loss, which measures the mean absolute error (MAE) between each element in the two input matrices. The epoch is set to 16, and the batch size is 2.

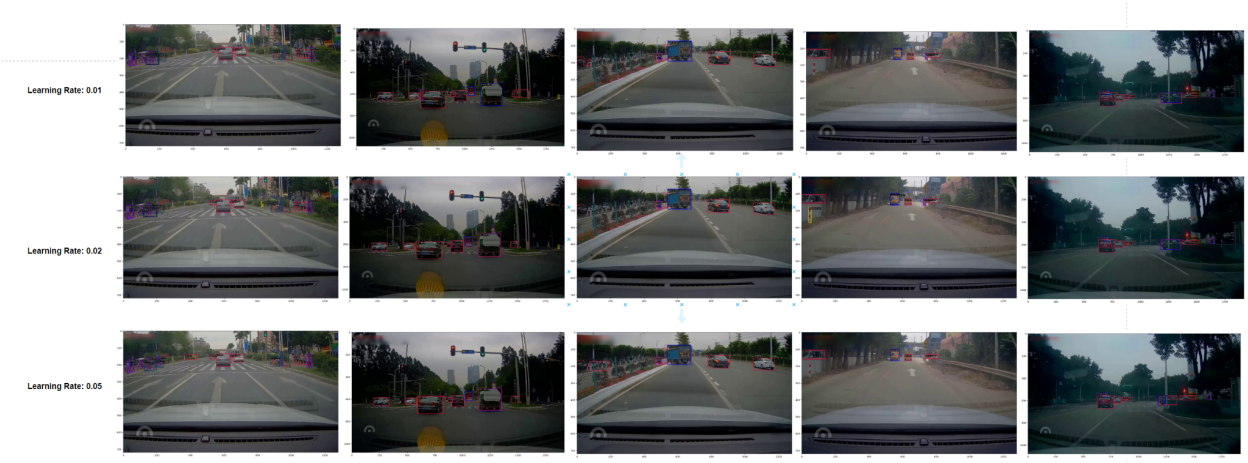
The learning rates tested are 0.02 and 0.05. In later sections when we change the optimizer to Adam, we will use a much smaller initial learning rate, otherwise the network will give un-ideal results with difficulties in convergence.

| Faster R-CNN | Config | | | | | | | Time | | Average Precision(AP) *[on validation set]* | | | | | | Average Recall(AR) *[on validation set]* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Batch Size | Epoch | BackBone | BottleNeck | Optimizer | Loss | BBox Loss | Train Time | Validation Time | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area |
| Learning Rate = 0.01 | 2 | 16 | ResNet50 | FPN | SGD | CE | L1 | 4h11min | 33s | 0.285 | 0.475 | 0.290 | 0.119 | 0.308 | 0.446 | 0.441 | 0.441 | 0.441 | 0.273 | 0.466 | 0.626 |
| Learning Rate = 0.02 | 2 | 16 | ResNet50 | FPN | SGD | CE | L1 | 4h46min | 35s | 0.285 | 0.476 | 0.294 | 0.123 | 0.310 | 0.451 | 0.447 | 0.447 | 0.447 | 0.278 | 0.472 | 0.647 |
| Learning Rate = 0.05 | 2 | 16 | ResNet50 | FPN | SGD | CE | L1 | 4h11min | 32s | 0.273 | 0.464 | 0.272 | 0.121 | 0.301 | 0.418 | 0.437 | 0.437 | 0.437 | 0.249 | 0.474 | 0.610 |




**Observations:**

1. Learning rate values tested are: 0.01, 0.02, 0.05; among which 0.02 has the best performances, with the highest AP and AR scores based on all IoU values.
2. Overall, current hyperparameter configuration does not produce promising results, we will next investigate how epoch amount and model components can impact the performances.
3. With all three learning rates, the model loss decreases massively then follows a gentle downward trend until iter 4000. Then with a small fluctuation the loss is pushed to another minima that is lower than the local minima before iter 4000. We should consider increasing the epoch amount to see if there exists a third bottom(minima), and whether it is better than the second local optimum.
4. Seen from the inference images, we notice that small area objects, especially blocks, are only detected out by the model with 0.02 learning rate, and this model's detection on small area objects are not as precise. On the contrary, all three models have comparatively better inference and higher confidence in detecting large-area objects or when the objects are unobstructed by other objects. Faster RCNN is used as a two-stage deep learning model for detecting these small objects; however, this model has some limitations in detecting small objects.

### 4.2 Anchor Scale

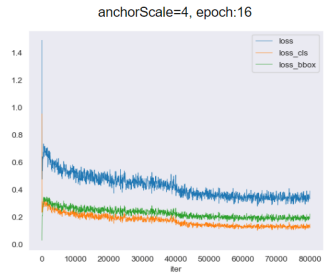
In practice, our object detection system will create a large number of “anchor boxes” to represent the location, shape and size of an object to be predicted. For each anchor box, we calculate which object’s bounding box has the highest Intersection Over Union(IOU).

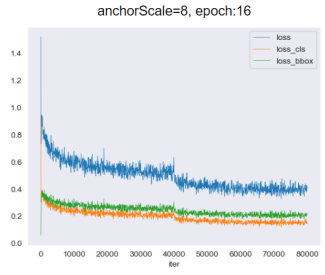
If the highest IOU is greater than 50%, tell the anchor box that it should detect the object that gave the highest IOU. Otherwise if the IOU is greater than 40%, tell the neural network that the true detection is ambiguous and not to learn from that example.  If the highest IOU is less than 40%, then the anchor box should predict that there is no object[10].

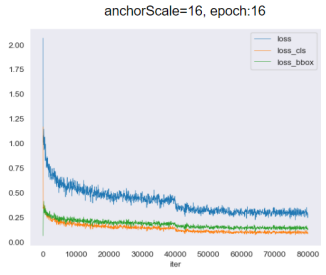
The size and the ratio of the anchor box determines the smallest/largest size box we want to be able to detect, and determines the shapes of objects to be detected. For example, a car detector might have short and wide anchor boxes.

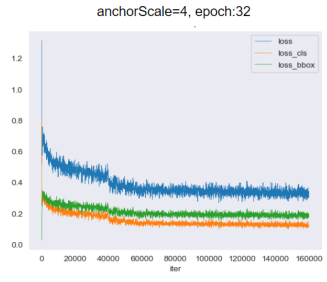
Due to the limitation of time budget, we will not run any parameter combinations with more than 16 epochs.  Keeping the anchor box ratio to [0.5, 1.0, 2.0], we tested box scales of 4, 8, 16.

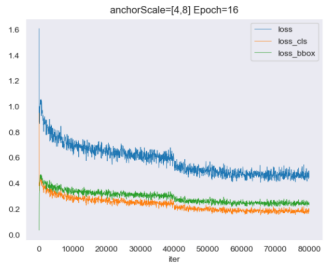
| Faster R-CNN | Config | | | | | | | | Time | | Average Precision(AP) *[on validation set]* | | | | | | Average Recall(AR) *[on validation set]* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Batch Size | Epoch | lr | BackBone | BottleNeck | Optimizer | Loss | BBox Loss | Train Time | Validation Time | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area |
| Anchor Scale = 4 | 2 | 16 | 0.02 | ResNet50 | FPN | SGD | CE | L1 | 4h14min | 34s | 0.276 | 0.463 | 0.282 | 0.115 | 0.303 | 0.444 | 0.444 | 0.444 | 0.444 | 0.271 | 0.460 | 0.614 |
| Anchor Scale = 4 | 2 | 32 | 0.015 | ResNet50 | FPN | SGD | CE | L1 | 8h26min | 33s | 0.279 | 0.468 | 0.285 | 0.120 | 0.301 | 0.446 | 0.453 | 0.453 | 0.453 | 0.269 | 0.473 | 0.622 |
| Anchor Scale = 8 | 2 | 16 | 0.02 | ResNet50 | FPN | SGD | CE | L1 | 4h46min | 35s | 0.285 | 0.476 | 0.294 | 0.123 | 0.310 | 0.451 | 0.447 | 0.447 | 0.447 | 0.278 | 0.472 | 0.647 |
| Anchor Scale = 16 | 2 | 16 | 0.01 | ResNet50 | FPN | SGD | CE | L1 | 5h26min | 32s | 0.234 | 0.383 | 0.241 | 0.031 | 0.245 | 0.448 | 0.343 | 0.343 | 0.343 | 0.041 | 0.371 | 0.626 |
| Anchor Scale = [4, 8] | 2 | 16 | 0.015 | ResNet50 | FPN | SGD | CE | L1 | 5h59min | 32s | 0.281 | 0.463 | 0.284 | 0.121 | 0.304 | 0.442 | 0.454 | 0.454 | 0.454 | 0.281 | 0.470 | 0.628 |

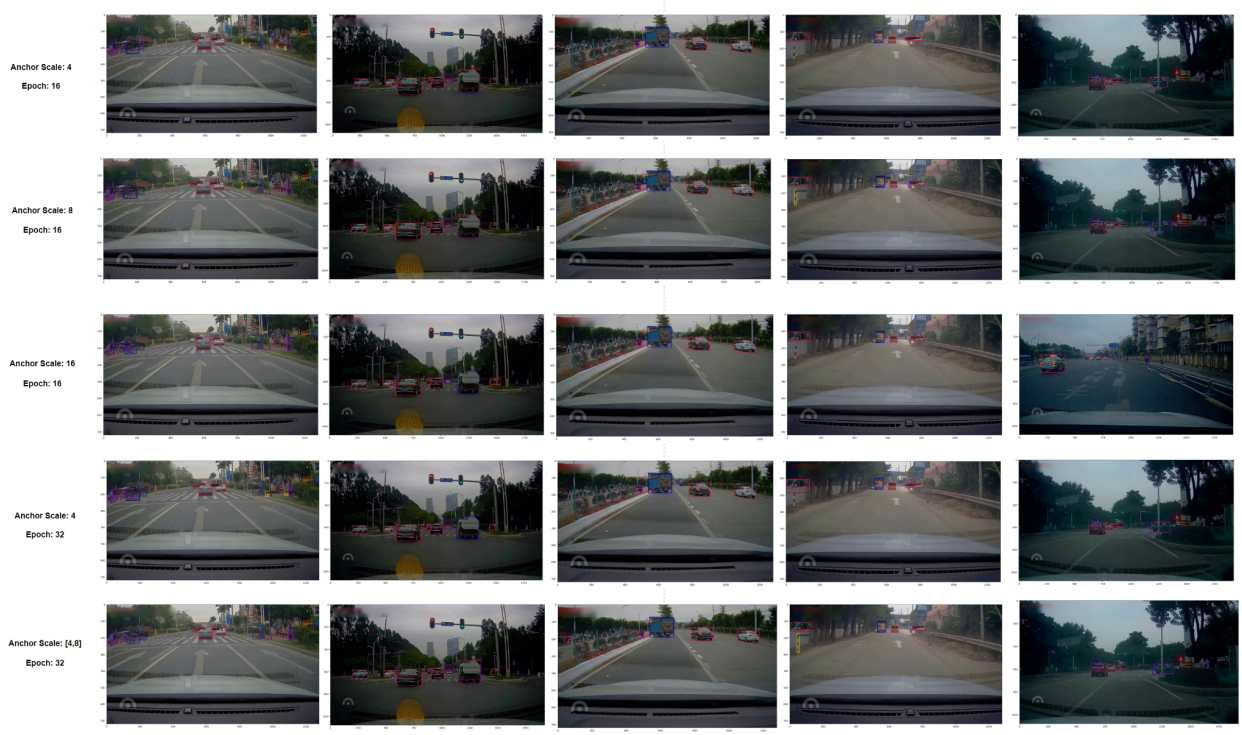





**Observations:**

1. The initial anchor scale is set to 8, and we can see from both the evaluation scores and the inference image(especially the images on the first column), that this scale can neglect the detection of pretty small objects. This is more obvious when the scale value increases to 16, that no small objects are detected out.
2. When we reduce the anchor scale, the model starts picking up small objects. However, for objects with larger size, misclassification happens. The truck is identified as a ‘car’ with two models of anchor scale 4.
3. During the training process, although not documented relative data here, we noticed that if a model has detected a small object, the detection can be neglected or cleaned up in the later stage of training, especially when overfitting happens or when the model fluctuates around the minima.
4. Since anchor scale 8 has shown limitations on small objects whilst 4 has downsides on correctly classifying larger objects, we also tested anchor scale [4, 8] to add intermediate anchor scales on each level.

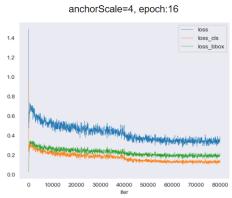
### 4.3 Bounding Box Loss

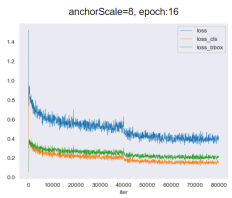
With complex background, occlusion and resolution not high enough, small object detection still has problems and limitations with Faster R-CNN. Inspired by the article “An Improved Faster R-CNN for Small Object Detection”[11], we compared the learning results of Faster R-CNN using L1 Loss and IoU loss.

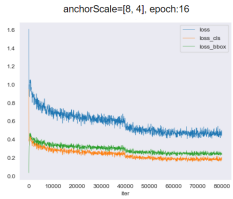
Intersection over Union(IoU) is the most popular evaluation metric used in object detection benchmarks. By using IoU loss, the training is a process trying to  maximize the Intersection over Union[12]. L1 on the contrary can be misleading when intersection happens.

| Faster R-CNN | Config | | | | | | | | Time | | Average Precision(AP) *[on validation set]* | | | | | | Average Recall(AR) *[on validation set]* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Batch Size | Epoch | lr | BackBone | BottleNeck | Optimizer | Loss | BBox Loss | Train Time | Validation Time | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area |
| Anchor Scale = 4 | 2 | 16 | 0.02 | ResNet50 | FPN | SGD | CE | L1 | 4h14min | 34s | 0.276 | 0.463 | 0.282 | 0.115 | 0.303 | 0.444 | 0.444 | 0.444 | 0.444 | 0.271 | 0.460 | 0.614 |
| Anchor Scale = 4 | 2 | 14 | 0.02 | ResNet50 | FPN | SGD | CE | IoU | 4h20min | 32s | 0.279 | 0.469 | 0.278 | 0.133 | 0.307 | 0.433 | 0.464 | 0.464 | 0.464 | 0.285 | 0.485 | 0.626 |
| Anchor Scale = 8 | 2 | 16 | 0.02 | ResNet50 | FPN | SGD | CE | L1 | 4h46min | 35s | 0.285 | 0.476 | 0.294 | 0.123 | 0.310 | 0.451 | 0.447 | 0.447 | 0.447 | 0.278 | 0.472 | 0.647 |
| Anchor Scale = 8 | 2 | 16 | 0.02 | ResNet50 | FPN | SGD | CE | IoU | 4h13min | 33s | 0.291 | 0.487 | 0.299 | 0.132 | 0.308 | 0.454 | 0.457 | 0.457 | 0.457 | 0.266 | 0.489 | 0.630 |
| Anchor Scale = [8, 4] | 2 | 16 | 0.015 | ResNet50 | FPN | SGD | CE | L1 | 5h59min | 32s | 0.281 | 0.463 | 0.284 | 0.121 | 0.304 | 0.442 | 0.454 | 0.454 | 0.454 | 0.281 | 0.470 | 0.628 |
| Anchor Scale = [8, 4] | 2 | 14 | 0.015 | ResNet50 | FPN | SGD | CE | IoU | 3h36min | 22s | 0.291 | 0.479 | 0.301 | 0.130 | 0.325 | 0.450 | 0.465 | 0.465 | 0.465 | 0.277 | 0.493 | 0.644 |

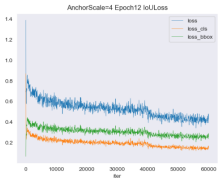
BBox Loss: L1

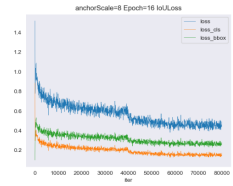


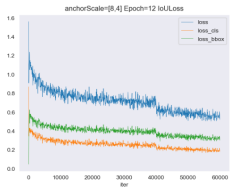




BBox Loss: IoU




**Anchor Scale: 4 Epoch: 16**
**L1 Loss**

**Anchor Scale: 4 Epoch: 16**
**IoU Loss**

**Anchor Scale: 8 Epoch: 16**
**L1 Loss**

**Anchor Scale: 8 Epoch: 16**
**IoU Loss**

**Anchor Scale: [8, 4] Epoch: 16**
**L1 Loss**

**Anchor Scale: [8, 4] Epoch: 16**
**IoU Loss**

**Observations:**

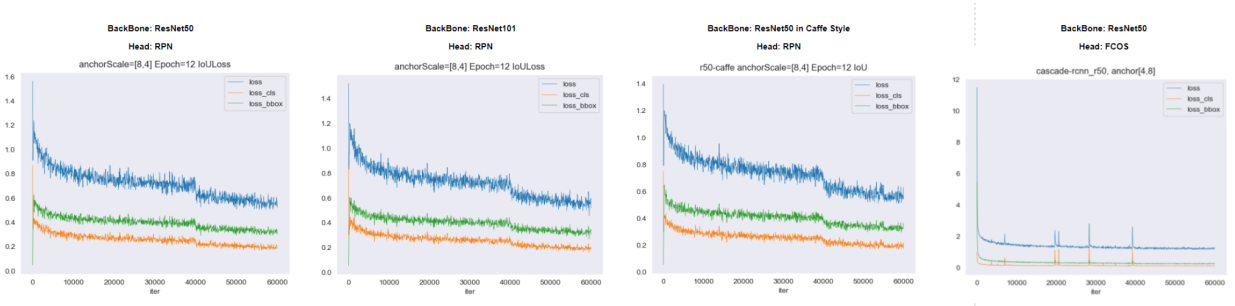
1. In Section 4.2, we noticed that a large anchor scale has better detection results on large objects such as cars and trucks. A small anchor scale has its advantages in identifying small objects for blocks. Therefore, we will test IoU loss with anchor scale 4 and 8 and compare the model performances. We could also notice from above that medium size objects such as person and bicycle can be more precisely detected and classified by anchor scale 8. This is the reason we also tested anchor scale [8, 4] in this section.
2. For anchor scale 4 and [8, 4], IoU loss has significantly improved the model performance(with much higher AP scores and UR scores). For anchor 8, only small and medium area objects detection is slightly enhanced, but overall, weaker than L1 loss. Overall, IoU Loss is proved to be slightly better than L1 loss.
3. The best AP and AR scores are produced by a Faster R-CNN model with IoU loss and anchor scale [8, 4]. It has comparatively better detection precision on small and medium objects. The detection on large objects is slightly weakened, but not too much. We will use this parameter combination in the later experiments. Also, see from the loss diagram, we could notice that the model may not finish convergence within the limited epoch. So we could expect an enhancement in this model with increased epoch amount.
4. In the inference images, small blocks detected by the model mentioned in Observation 3 are weaker than the model with the same anchor scale but with L1 Loss. One assumption is that the model has not fully converged.

## 4 Different Components for Faster R-CNN

VGG16 and ResNet50 are two commonly used backbone structures for Faster R-CNN. Due to the mmdetection module not supporting VGG16 for Faster R-CNN and the time budget, we sadly do not test how VGG16 can affect the learning results. In this assignment, we tested three different backbones: ResNet50, ResNet101 and ResNet50 in Caffe style. We also switched the head of the Faster R-CNN model from RPN to FCOS.

FCOS is originally proposed as a fully convolutional one-stage object detector to solve object detection in a per-pixel prediction fashion, analog to semantic segmentation[14]. It is designed deliberately to avoid complicated computation related to anchor boxes such as calculating overlapping during training. The reason we can use it as RPN is because bounding box predictions by FCOS can also be regarded as region proposals[15].

| Faster R-CNN | Config | | | | | | | | | Time | | Average Precision(AP) *(on validation set)* | | | | | | Average Recall(AR) *(on validation set)* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Batch Size | Epoch | lr | BackBone | BottleNeck | RPN Head | Optimizer | Loss | BBox Loss | Train Time | Validation Time | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area | AR @ IoU = 0.5:0.95 All Area | AR @ IoU = 0.5 All Area | AR @ IoU = 0.75 All Area | AR @ IoU = 0.5:0.95 Small Area | AR @ IoU = 0.5:0.95 Medium Area | AR @ IoU = 0.5:0.95 Large Area |
| Anchor Scale = [8, 4] | 2 | 12 | 0.02 | ResNet50 | FPN | RPN Head | SGD | CE | IoU | 3h36min | 22s | 0.291 | 0.479 | 0.301 | 0.130 | 0.325 | 0.450 | 0.465 | 0.465 | 0.465 | 0.277 | 0.493 | 0.644 |
| Anchor Scale = [8, 4] | 2 | 12 | 0.02 | ResNet101 | FPN | RPN Head | SGD | CE | IoU | 4h28min | 41s | 0.277 | 0.459 | 0.279 | 0.122 | 0.304 | 0.445 | 0.445 | 0.445 | 0.445 | 0.261 | 0.470 | 0.629 |
| Anchor Scale = [8, 4] | 2 | 12 | 0.02 | ResNet50-Caffe | FPN | RPN Head | SGD | CE | IoU | 3h22min | 39s | 0.279 | 0.468 | 0.279 | 0.133 | 0.301 | 0.443 | 0.460 | 0.460 | 0.460 | 0.285 | 0.485 | 0.624 |
| Anchor Scale = [8, 4] | 2 | 12 | 0.0001 | ResNet50 | FPN | FCOS | SGD | CE | IoU | 3h15min | 35s | 0.172 | 0.325 | 0.157 | 0.035 | 0.176 | 0.328 | 0.311 | 0.311 | 0.311 | 0.091 | 0.323 | 0.533 |





**Observations:**

1. Models with RPN head have similar convergence curve trends, RPN-FCOS on the contrary, has a much different trend. The Faster R-CNN model with RPN-FCOS converged really fast, but still leaving the loss the highest among the four models tested. This suggests that we are probably facing an exploding gradients problem. We should try to use a much lower learning rate for better performance.
2. For the three models with RPN head, ResNet50 in Caffe style has the highest loss. We can also see this from the AP and AR diagram above. ResNet50 in Caffe style has weaker performances compared to the torch version ResNet50. It has its advantage in identifying small objects, but fails to carry out good detection on larger targets. For example, in the inference images, the model mistakenly identifies certain trucks and vans to cars.
3. The model with ResNet101 backbone overall is the second best performance. Although not having highest evaluation scores across all areas or IoU ranges, it has more even performances on different size objects detection. The loss with this model is the second lowest, only slightly higher than the ResNet50 backbone.
4. The Faster R-CNN model with ResNet101 backbone shows a better capability in the field of detecting small objects from varied backgrounds.
5. Faster R-CNN with ResNet50 has the best performance, and weaker performance on small object detection compared to ResNet101. However, theoretically, ResNet101 should be better than ResNet50 in general terms. Our experiment shows the opposite results potentially because:

a. Hyperparamter settings do not find the optimal solution;
b. It is easy for ResNet101 to overfit.

6. Faster R-CNN with ResNet50 has the best performance, and weaker performance on small object detection compared to ResNet101. However, theoretically, ResNet101 should be better than ResNet50 in general terms. Our experiment shows the opposite results potentially because:

## 5 Other Object Detection Networks

We choose Faster R-CNN with RPN head and ResNet50 backbone, the best performing model from Section 4, as a baseline to compare with other detection networks. Cascade R-CNN and Dynamic R-CNN from the R-CNN family are validated in this section.

### 5.1 Cascade R-CNN

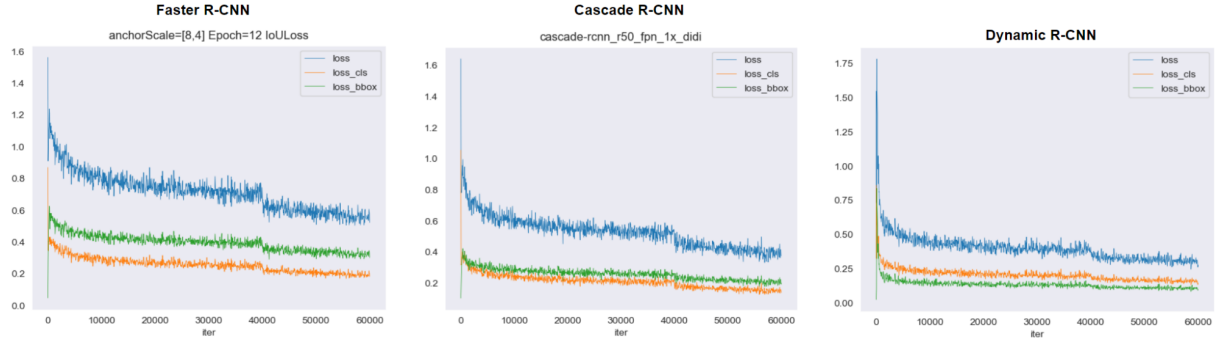
Cascade R-CNN is an object detection architecture that seeks to address problems with degrading performance with increased IoU thresholds (due to overfitting during training and inference-time mismatch between IoUs for which detector is optimal and the inputs)[16].

### 5.2 Dynamic R-CNN

Dynamic R-CNN is an object detection method that adjusts the label assignment criteria (IoU threshold) and the shape of regression loss function (parameters of Smooth L1 Loss) automatically based on the statistics of proposals during training. The motivation is that in previous two-stage object detectors, there is an inconsistency problem between the fixed network settings and the dynamic training procedure. For example, the fixed label assignment strategy and regression loss function cannot fit the distribution change of proposals and thus are harmful to training high quality detectors[17].

### 5.3 Experiment Diagrams

| | Config | | | | | | | | Time | | Average Precision(AP) *(on validation set)* | | | | | | Average Recall(AR) *(on validation set)* | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Batch Size | Epoch | lr | BackBone | BottleNeck | Optimizer | Loss | BBox Loss | Train Time | Validation Time | AP @ IoU = 0.5:0.95 All Area | AP @ IoU = 0.5 All Area | AP @ IoU = 0.75 All Area | AP @ IoU = 0.5:0.95 Small Area | AP @ IoU = 0.5:0.95 Medium Area | AP @ IoU = 0.5:0.95 Large Area | AR @ IoU = 0.5:0.95 All Area | AR @ IoU = 0.5 All Area | AR @ IoU = 0.75 All Area | AR @ IoU = 0.5:0.95 Small Area | AR @ IoU = 0.5:0.95 Medium Area | AR @ IoU = 0.5:0.95 Large Area |
| Faster R-CNN Anchor Scale = [8, 4] | 2 | 12 | 0.02 | ResNet50 | FPN | SGD | CE | loU | 3h36min | 22s | 0.291 | 0.479 | 0.301 | 0.130 | 0.325 | 0.450 | 0.465 | 0.465 | 0.465 | 0.277 | 0.493 | 0.644 |
| Dynamic R-CNN | 2 | 12 | 0.02 | ResNet50 | FPN | SGD | CE | Smooth L1 | 3h38min | 33s | 0.278 | 0.437 | 0.290 | 0.119 | 0.299 | 0.446 | 0.430 | 0.430 | 0.430 | 0.236 | 0.451 | 0.625 |
| Cascade R-CNN | 2 | 12 | 0.02 | ResNet50 | FPN | SGD | CE | loU | 4h12min | 41s | 0.294 | 0.471 | 0.307 | 0.135 | 0.315 | 0.465 | 0.457 | 0.457 | 0.457 | 0.272 | 0.478 | 0.643 |




**Observations:**

1. Cascade R-CNN has the worst performance among the three networks we tested. However, it shows good capability of detecting small size objects. The Cascade R-CNN proposes a fix for challenges and phenomenons in the IoU overlap threshold problem. The problem is: with small IoU threshold results in noisy samples, and training with large thresholds degrades the performance because of overfitting and low-quality region proposals at inference compared to training[18]. The fix relies on a technique known as “ROIAlign”, and is crucial for instance segmentation. This network has its limitations on giving low confidence with high probability to the RoIs whose IoU is between 0.5 and 0.7 and will not regress these RoIs[19].
2. Dynamic R-CNN has good AP when the IoU is comparatively high(IoU=0.75). In fact, compared to Faster R-CNN, AP scores with Dynamic R-CNN seem to produce more reliable results. However, its AR scores are generally lower than Faster R-CNN. AP(in precision) measures the ratio of the number of true positives to the total number of positive predictions. And AR(in recall) evaluates the ratio of the number of true positives to the total number of actual (relevant) objects. For a fine-tuned model, we need both high precision scores and recall scores. As the diagram shows, we could possibly say, that on our dataset and training pipeline, if Dynamic R-CNN can detect something out as a labeled object, it has good reliability that this object is correctly classified, however, it has comparatively weak capability yielding out a proposal is an object.
3. Faster R-CNN still seems to be the most reliable network amongst the ones we tested. It has better inference to detect an object out, but slightly not that good on classifying the object to the ground truth class.

## 6 Summary

In our experiment, we tested how backbone networks and heads can affect the behavior of the Faster R-CNN model. We also investigate the impact of anchor box size(scale) in the field of detecting objects of different sizes in varied backgrounds. Two other models from the R-CNN family, Dynamic R-CNN and Cascade R-CNN, are compared to the Faster R-CNN.

With anchor scale set to [4, 8] and using IoU Loss as the bounding box loss, Faster R-CNN has the best Average Precision and Average Recall scores among the models we tested, under a limited epoch amount. When IoU is 0.5, the AP is 0.479 and when IoU is 0.75, the AP is 0.301. For IoU in the range of 0.5 to 0.95, the AP reaches 0.291. This model has the highest AP 0.325 for medium size object

detection. For small and large size objects, the APs are 0.130 and 0.450. Although most of its AP scores are not the highest, it is still one of the best, together with high AR scores acrossing all IoU ranges.

If time applies, it would be worth a try to see how non-RCNN family models behave on our dataset. It would also be interesting to dig in deeper to all networks we tested, especially when changing the backbone/head/neck for Faster R-CNN. The Cascade R-CNN and Dynamic R-CNN have comparatively weaker performances compared to how some materials suggest. We should run more tests and tunings to validate and analyze this phenomenon.

## 7 References


[1] Object Recognition – 3 things you need to know
https://www.mathworks.com/solutions/image-video-processing/object-recognition.html#:~:text=Object%20detection%20is%20the%20process,located%20within%20the%20same%20image.

[2] Top 8 Algorithms For Object Detection
https://analyticsindiamag.com/top-8-algorithms-for-object-detection/

[3] D2City https://paperswithcode.com/dataset/d2city

[4] coco - Common Objects in Context https://cocodataset.org/#home

[5] mAP (mean Average Precision) for Object Detection
https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[6] OpenMMLab Detection Toolbox and Benchmark https://paperswithcode.com/lib/mmdetection

[7] Faster R-CNN for object detection
https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46

[8] Review: FPN — Feature Pyramid Network (Object Detection)
https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610

[9] A robust approach for industrial small-object detection using an improved faster regional convolutional neural network
https://www.nature.com/articles/s41598-021-02805-y#:~:text=Faster%20RCNN%20is%20used%20as,%2C%20untighten%20screw%2C%20and%20labels.

[10] Anchor Boxes — The key to quality object detection
https://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9

[11] An Improved Faster R-CNN for Small Object Detection https://ieeexplore.ieee.org/document/8786135

[12] Bounding Box Regression Loss
https://wiki.cloudfactory.com/docs/mp-wiki/loss/bounding-box-regression-loss

[13] What is the difference between using VGG or ResNet for transfer learning in CNN models? https://www.quora.com/What-is-the-difference-between-using-VGG-or-ResNet-for-transfer-learning-in-CNN-models

[14] Tian, Z. et al. (2019) 'FCOS: Fully convolutional one-stage object detection', 2019 IEEE/CVF International Conference on Computer Vision (ICCV) [Preprint]. doi:10.1109/iccv.2019.00972.

[15] USE A SINGLE STAGE DETECTOR AS RPN https://mmdetection.readthedocs.io/en/3.x/user_guides/single_stage_as_rpn.html

[16] Cai, Z. and Vasconcelos, N. (2021) 'Cascade R-CNN: High Quality Object Detection and instance segmentation', IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(5), pp. 1483–1498. doi:10.1109/tpami.2019.2956516.

[17] Zhang, H. et al. (2020) 'Dynamic R-CNN: Towards High Quality Object Detection via dynamic training', Computer Vision – ECCV 2020, pp. 260–275. doi:10.1007/978-3-030-58555-6_16.

[18] Fundamentals of Object Detection: Faster, Mask, Cascade R-CNN https://medium.com/mlearning-ai/details-of-faster-mask-cascade-r-cnn-7b9c34326cdf

[19] Rethinking Classification and Localization for Cascade R-CNN https://arxiv.org/pdf/1907.11914