# Assignment 1 report

Wei Fuyang

March 9, 2023

## 1 Background

### 1.1 Object Detection

Image classification, detection and segmentation are three major tasks in the field of computer vision. The image classification model divides the image into a single category, usually corresponding to the most prominent object in the image. However, many images in the real world usually contain more than one object. At this time, it is very rough and inaccurate to use the image classification model to assign a single label to the image. In this case, a target detection model is needed. The target detection model can identify multiple objects in a picture and locate different objects (give the boundary box).

### 1.2 YOLO

YOLO solves object detection as a regression problem. Based on a separate end-to-end network, it completes the input from the original image to the output of the object position and category.

The YOLO detection network includes 24 convolution layers and 2 full connection layers. The convolution layer is used to extract image features, and the full connection layer is used to predict image position and category probability values.

The original YOLO network drew on the GoogLeNet classification network structure. The difference is that YOLO does not use the concept module, but uses 1x1 volume layer (where 1x1 volume layer exists for cross-channel information consolidation)+3x3 volume layer for simple replacement.

It is noteworthy that in our experiment, we replaced the convolutional network with Resnet50.

## 2 Experiment

### 2.1 Default Settings

After completing all the codes, we first use the default hyper-parameter settings to train the model. The specific parameter settings are as follows:

| Parameters | Value |
|---|---|
| Input size | 448 |
| Epochs | 10 |
| Batch size | 12 |
| Learning Rate | 5e-5 |

After training the model in the default settings, we can evaluate the model on the validation set. The evaluation mertics are as follows:

| Class | AP |
|---|---|
| Pedestrian | 8.11 |
| Cyclist | 30.63 |
| Car | 56.67 |
| Truck | 29.58 |
| Tram | 25.00 |

The Mean Average Precision (mAP) is 0.30.

## 2.2 Improvement

We find that under the default parameter setting, the decline of loss will stop prematurely. We speculate that it may be due to the complexity of the loss function that leads to the local extremum in the process of optimization.

To solve this problem, we try to add weight decay in the optimizer and introduce learning rate decay mechanism in the training process.

### 2.2.1 Optimizer Weight Decay

To put it simply, we add weight decay to punish the optimized parameters, which is a very basic skill in machine learning and deep learning. In this experiment, we use the AdamW optimizer, and we use the parameter *weight decay* is set to 5e-4.

### 2.2.2 Learning Rate Decay

The learning rate can control the pace of renewal. When we train the model, the learning rate will be relatively high at the beginning, so that we can reach the optimal value at a relatively fast speed, and then reduce the learning rate, slowly converge to the optimal value.

In this experiment, we adopt adaptive learning rate decay, which means that the learning rate will decay only when the loss does not decline for a period of time.

In Pytorch, we use the function ReduceLROnPlateau to implement adaptive decay of learning rate.

We set patience=0 (which means we check the loss in every epoch) and factor=0.85 (which means the new learning rate is 0.85 of the old learning rate).

## 2.3 Result

After improvement, we reset the hyper parameters as follow:

| Parameters | Value |
|---|---|
| Input size | 448 |
| Epochs | 60 |
| Batch size | 8 |
| Initial Learning Rate | 0.0001 |

### 2.3.1 Loss and Learning Rate curve

After training, we plot the curve which is Loss and LR vs Epochs:

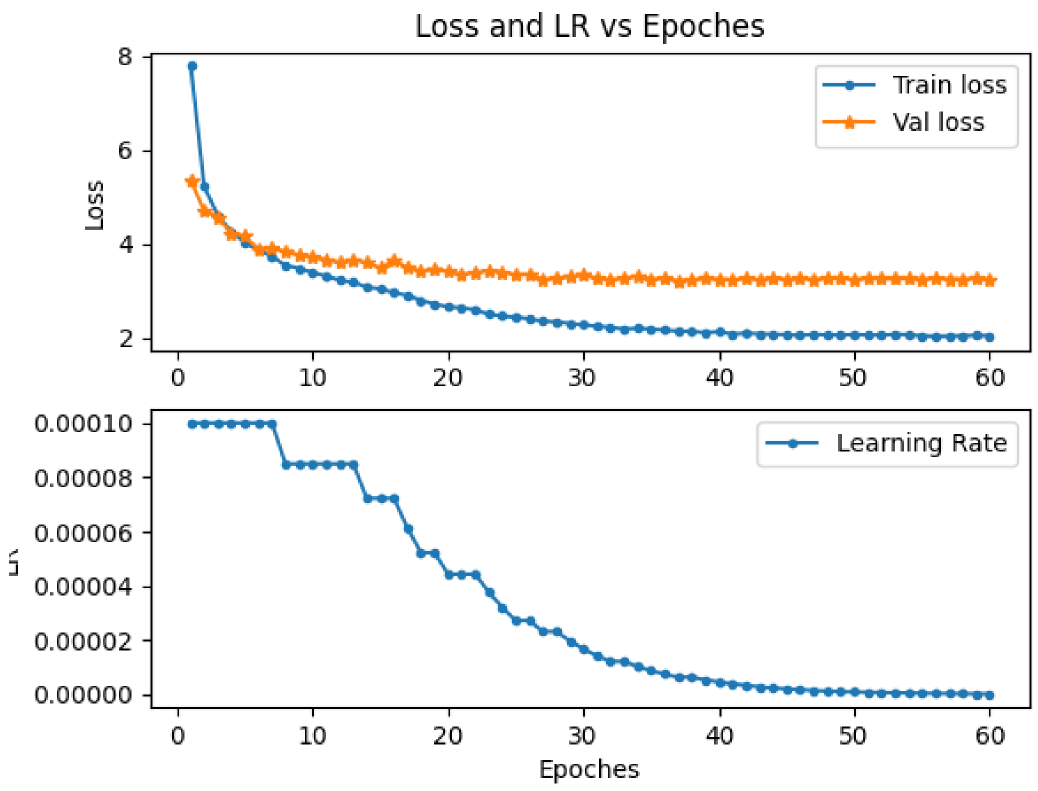

Figure 1: Loss and LR vs Epochs

### 2.3.2 New validation metrics

| Class | AP |
|---|---|
| Pedestrian | 23.63 |
| Cyclist | 40.65 |
| Car | 67.89 |
| Truck | 48.38 |
| Tram | 36.05 |

The new Mean Average Precision (mAP) is 0.43.

### 2.3.3 Prediction

We apply the model on test dataset and get some prediction pictures:

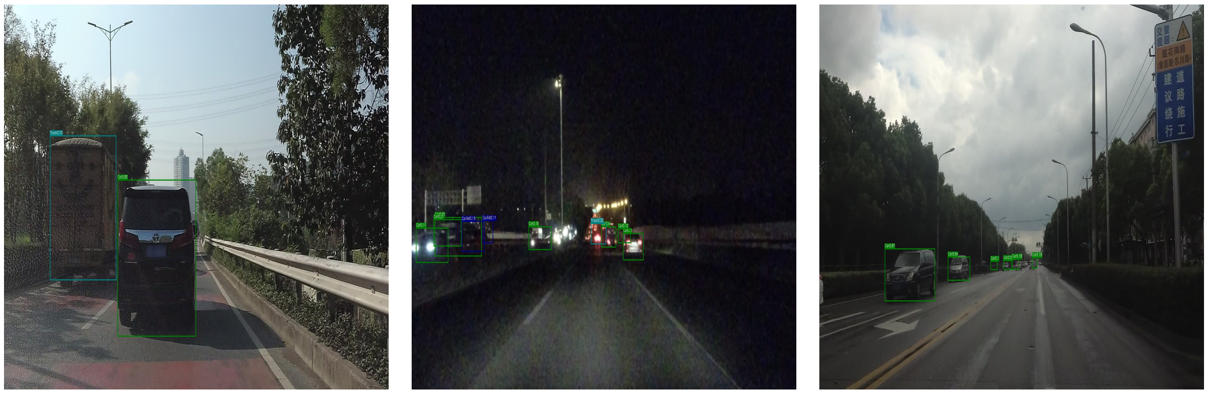
Figure 2: Prediction

## 3 Discussion

Under the default hyperparameter setting, we found that the performance of the model did not meet our expectations, and we could not significantly improve the performance under the conventional parameter adjustment. Therefore, we choose to introduce the weight decay of the optimizer and learning rate decay.We found that the performance of the model was indeed improved after the introduction of these techniques.

In fact, we can also continue to optimize the training process. Looking at Figure 1, we can see that after 30 epochs, the learning rate continues to decline while the validation error is almost stable. We can think that the attenuation of the learning rate has not been able to further optimize the model. We speculate that the performance of the model has not improved after 30 epochs, so we can only train 30 to 40 epochs to reduce the training cost.