Assignment Task 2: Project Implementation

Project Direction: Object Detection

Team 4

Weilin Sun 13462383
Yangyang Jin 13647716
Junting Song 13833936
Qibai Chen 13757467
Tianyang Zhang 13653112

Table of Contents

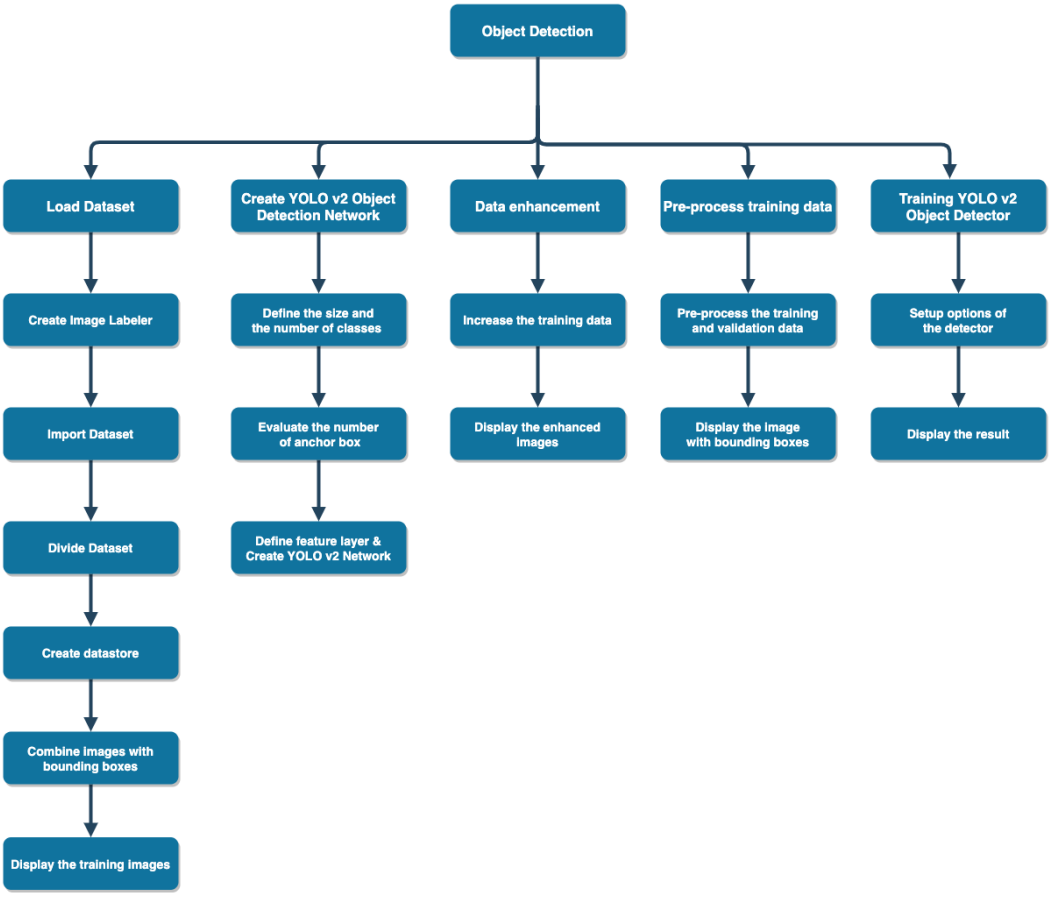
1. Introduction

In the previous Project Requirements and Specifications (Assessment Task 1), we described what object detection is and the availability of our chosen dataset. Through the knowledge we gained in independent reading and lectures, we found two more common object detection technologies and also carried out in-depth descriptions of these two technologies, which are RCNN and YOLO. However, we finally chose the YOLOv2 object detection network for our project. In this report, we will give a detailed description of the code operation instructions, software development process and evaluation methods. At the end we will also discuss the problems and challenges encountered in this project.

2. High level description of the code

2.1 Flowcharts

In this project, the realization of object detection technology is mainly divided into five parts, which are loading the dataset, creating YOLOv2 object detection network, data enhancement, preprocessing the training data, and training the YOLOv2 Object detector.



In the loading data set, we first need to create an Image Labeler, which is to mark the location of the vehicle in the training set. Then we need to separate the data set and divide it into training set, validation set and test set. After that, we need to create a

datastore and combine the image and bounding box together. Finally, the training set image can be displayed.

The second part is to create the YOLO object detection network. In this part, we need to redefine the image size, evaluate the number of anchor boxes, and define the feature layer. Next is data enhancement. Its purpose is to improve the accuracy of the network by randomly transforming the original data during training.

The fourth part is preprocessing the training data. Finally, we can train the YOLO v2 target detector and test the results.

2.2 Instructions for operation of the code

In this part, the function and application of the code will be explained in detailed.

2.21 Load Dataset

1) Create Image Labeler

Firstly, we need to use the built-in application Image Labeler in MATLAB to import the training set images and create a label name. As shown in the figure below (refer to Figure 1), we need to manually draw the bounding box of the vehicle to show the position of the vehicle in the image. Finally, export these images with bounding boxes to the workspace as gTruth. The purpose of this step is to let the training set know where the vehicle is and provide a standard "answer" for the training set.

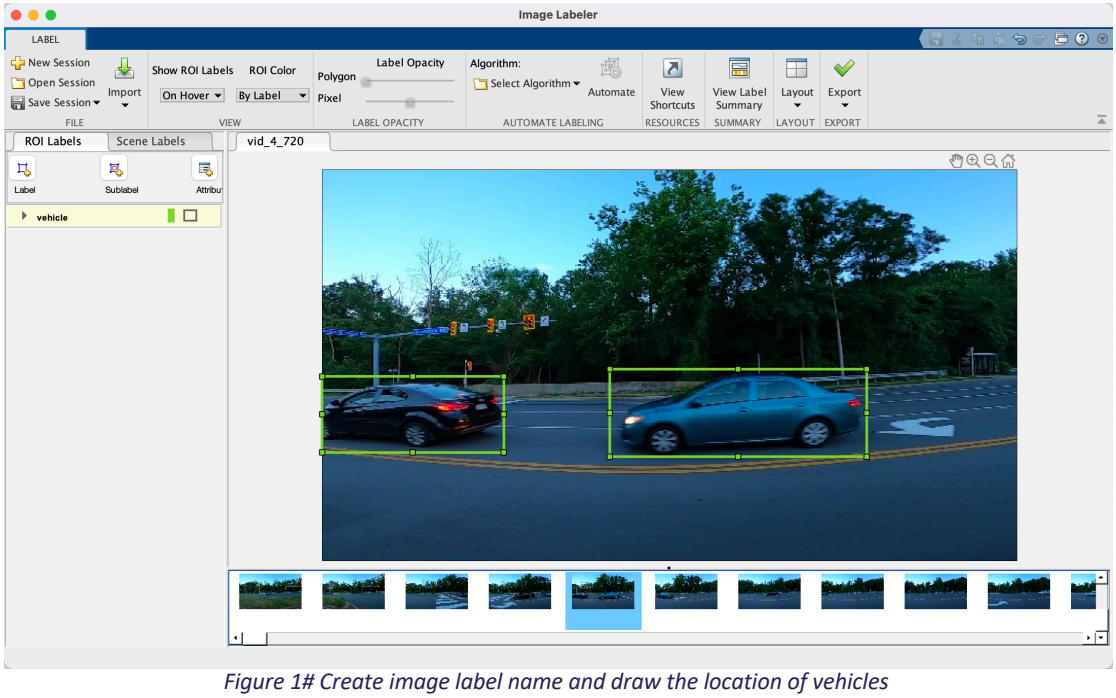


Figure 1# Create image label name and draw the location of vehicles

2) Import Dataset

The next step is to load the data set. As shown in the figure below (refer to Figure 2), the vehicle data is stored in a two-column table. The first column contains the image file path, and the second column contains the vehicle bounding box.

```
% Import dataset
imageFilename = gTruth.DataSource.Source;
vehicleDataset = table(imageFilename);
vehicleDataset = cat(2,vehicleDataset ,gTruth.LabelData);
vehicleDataset(1:4,:)
```

```
ans =

  4×2 table
```

imageFilename	vehicle
{'C:\Users\11566\Desktop\Assignment2\Assignment 2\car\training_images\vid_4_2000.jpg'}	{4×4 double}
{'C:\Users\11566\Desktop\Assignment2\Assignment 2\car\training_images\vid_4_1800.jpg'}	{3×4 double}
{'C:\Users\11566\Desktop\Assignment2\Assignment 2\car\training_images\vid_4_1820.jpg'}	{3×4 double}
{'C:\Users\11566\Desktop\Assignment2\Assignment 2\car\training_images\vid_4_1840.jpg'}	{2×4 double}

Figure 2# Training set image path and vehicle bounding box information

3) Divide Dataset

After loading the data set, we divide the data set into three parts: training set, validation set and test set. Among them, the training set accounts for 60%, the validation set accounts for 10%, and the remaining 30% is the test set.

The rng(0) stands for random number generator which means the generator using a seed of 0. By using the randperm, it will confer an index to the all the data randomly to avoid overfitting.

```
rng(0);
shuffledIndices = randperm(height(vehicleDataset));
idx = floor(0.6 * length(shuffledIndices) );

trainingIdx = 1:idx;
trainingDataTbl = vehicleDataset(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = vehicleDataset(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = vehicleDataset(shuffledIndices(testIdx),:);
```

4) Create datastore

Then, we need to convert the address to a picture, use imageDatastore to get the picture, and boxLabelDatastore to get the bounding box label. Line 23 will give the path to save the training data image, and Line 24 will save the location of the ground frame and its label as a vehicle. In addition, the next few lines of code will play the same role, for validation sets and test sets, respectively.

```
23 -    imdsTrain = imageDatastore(trainingDataTbl{:,'imageFilename'});
24 -    bldsTrain = boxLabelDatastore(trainingDataTbl(:,'vehicle'));
25
26 -    imdsValidation = imageDatastore(validationDataTbl{:,'imageFilename'});
27 -    bldsValidation = boxLabelDatastore(validationDataTbl(:,'vehicle'));
28
29 -    imdsTest = imageDatastore(testDataTbl{:,'imageFilename'});
30 -    bldsTest = boxLabelDatastore(testDataTbl(:,'vehicle'));
```

5) Combine images with bounding boxes

After converting the image path to a picture, the next step is to combine the image and the bounding box together so that they can appear in one picture.

```
trainingData = combine(imdsTrain,bldsTrain);
validationData = combine(imdsValidation,bldsValidation);
testData = combine(imdsTest,bldsTest);
```

6) Display the training images and bounding box labels

Finally, we can display the training image and its corresponding bounding box label (Refer to Figure 3).

```
data = read(trainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,'Rectangle',bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)
```

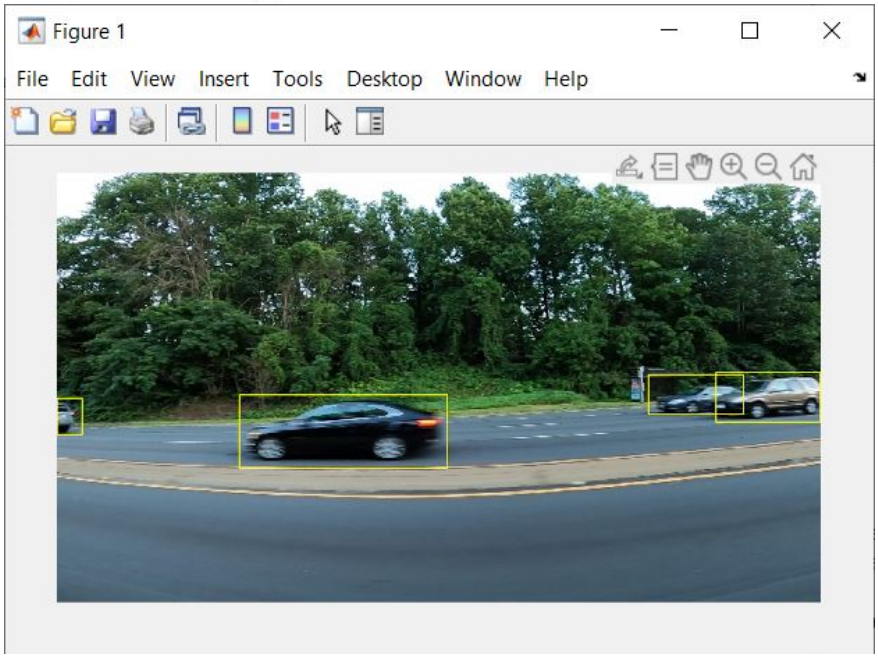


Figure 3# Display the training image and its corresponding bounding box label

2.22 Create YOLO v2 Object Detection Network

1) Define the size and the number of classes

Firstly, in order to reduce the computational cost of training, we set the network input size to [224 224 3], which is the minimum size required to run the network. Then, numClasses function will identify the number of classes of the object by using the number of columns in the table.

```
inputSize = [224 224 3];
numClasses = width(vehicleDataset)-1;
```

2) Evaluate the number of anchor box

```
trainingDataForEstimation = transform(trainingData,@(data)preprocessData(data,inputSize));
numAnchors = 7;
[anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingDataForEstimation, numAnchors);
```

Then, the estimateAnchorBoxes will be used to predict the anchor boxes for deep learning object detectors. To accommodate for the resizing of the photographs before training, resize the training data for calculating anchor boxes. Transform the training data first, then define and estimate the number of anchor boxes. Resize the training data to the input image scale of the network using the accompanying function preprocessData. The percent of IoU is a measurement of the true-to-predicted correlation, the stronger the correlation, the higher the number.

3) Define feature layer & Create YOLO v2 Network

```
featureLayer = 'activation_40_relu';
lgraph = yolov2Layers(inputSize,numClasses,anchorBoxes,resnet50,featureLayer);
```

Based on the code the feature layer will be defined as the 'activation_40_relu' will be used to instead the layers with the detection subnetwork which can generates down sampled about 16 times. In this case, it can make a balance between spatial resolution and the extracted features' strength. Furthermore, the YOLO v2 Object Detection Network will be create with input size, the number of classes and anchor boxes, feature extraction network, which is resnet50, and the feature layer.

2.23 Data enhancement

1) Increase the training data

```
augmentedTrainingData = transform(trainingData,@augmentData);
```

Data enhancement improves network accuracy by randomly transforming raw data during training. By using data enhancement, we can add more variations to the training data without increasing the number of labeled training samples.

In this part, the transform function will be used to strengthen the training data by randomly rotating the picture and accompanying bounding box labels horizontally which can improve the accuracy of the network at the same time.

2) Display the enhanced images

```
augmentedData = cell(4,1);
for k = 1:4
    data = read(augmentedTrainingData);
    augmentedData{k} = insertShape(data{1},'Rectangle',data{2});
    reset(augmentedTrainingData);
end
figure
montage(augmentedData,'BorderSize',10)
```

In this case, A for loop will be used to read the same image multiple times. The following figure displays the image which has been augmented (Refer to Figure 4).

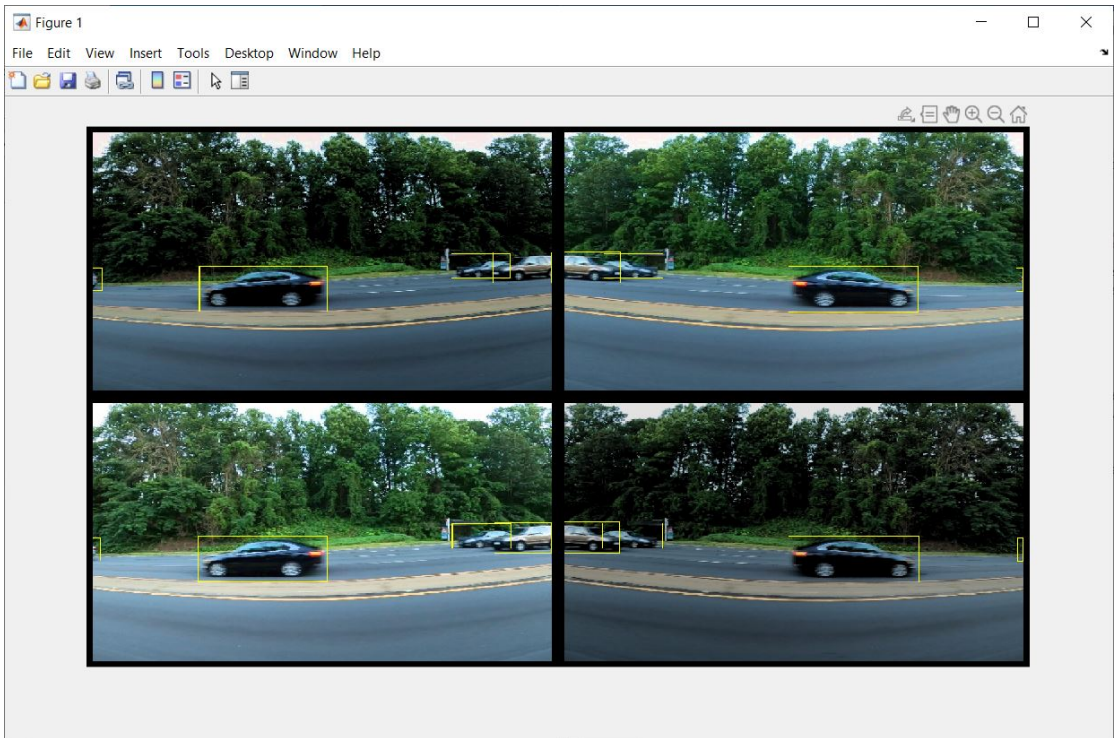


Figure 4# Image enhanced using Transform

2.24 Pre-process training data

1) Pre-process the training and validation data

```
74 -    preprocessedTrainingData = transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
75 -    preprocessedValidationData = transform(validationData,@(data)preprocessData(data,inputSize));
76
77 -    data = read(preprocessedTrainingData);
```

Use transform function to pre-process both training and validation data, then read the pre-processed data.

2) Display the image with bounding boxes

```
79 -    I = data{1};
80 -    bbox = data{2};
81 -    annotatedImage = insertShape(I,'Rectangle',bbox);
82 -    annotatedImage = imresize(annotatedImage,2);
83 -    figure
84 -    imshow(annotatedImage)
```

This part is to display the image after preprocessing (Refer to Figure 5).

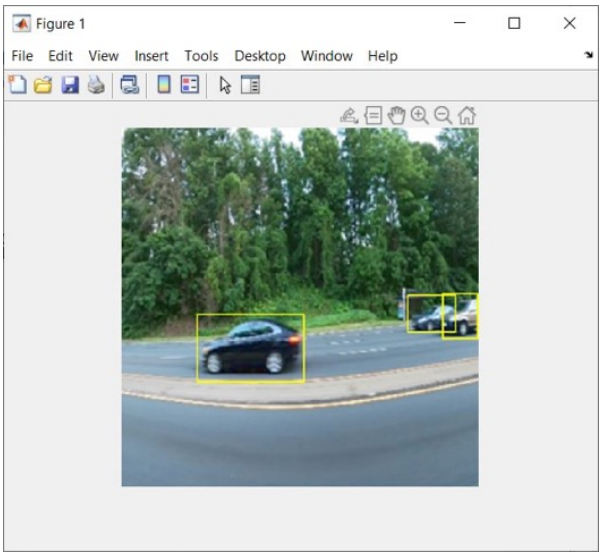


Figure 5# Image after preprocessing

2.25 Training YOLO v2 Object Detector

1) Setup options of the detector

```
87 -    options = trainingOptions('sgdm', ...
88              'MiniBatchSize',16, ....
89              'InitialLearnRate',1e-3, ...
90              'MaxEpochs',8, ...
91              'CheckpointPath',tempdir, ...
92              'ValidationData',preprocessedValidationData,...
93              'Plots','training-progress');
94          %layer = detector.Network.Layers;
95 -     [detector,info] = trainYOLOv2ObjectDetector(preprocessedTrainingData,lgraph,options);
```

The code of training YOLO detector has been displayed above. Based on the code, the trainingOptions is used to indicate the settings for network training. Furthermore, tempdir means the temporary directory. Set 'CheckpointPath' as a temporary point to save some of trained detectors while processing, then set up 'ValidationData' as pre-processed validation data. Finally, train the detector by using trainYOLOv2ObjectDetector function which will use pre-processed training data, layer graph, and options between line 87 and line 93. The result page will be displayed below (Refer to Figure 7&8).

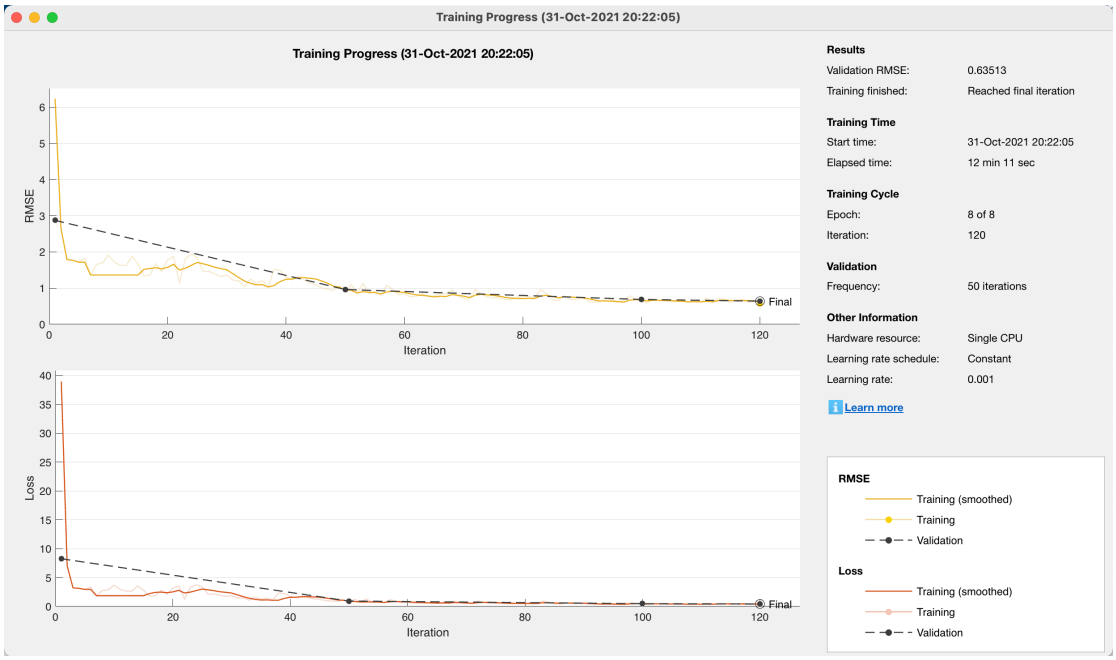


Figure 6# Training process

```
* vehicle

Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================|
|       1 |           1 |       00:00:23 |         6.24 |         2.87 |      38.9270 |       8.2483 |          0.0010 |
|       4 |          50 |       00:05:14 |         0.88 |         0.96 |       0.7761 |       0.9238 |          0.0010 |
|       7 |         100 |       00:09:59 |         0.63 |         0.69 |       0.4029 |       0.4711 |          0.0010 |
|       8 |         120 |       00:11:59 |         0.56 |         0.64 |       0.3109 |       0.4090 |          0.0010 |
|=========================================================================================================================|
Detector training complete.
*************************************************************************
```

Figure 7# Training result information

2) Display the result

```
 96 -    I = imread('vid_4_2280.jpg');
 97 -    imshow(I);
 98 -    I = imresize(I,inputSize(1:2));
 99 -    [bboxes,scores] = detect(detector,I);
100 -    I = insertShape(I,'Rectangle',bboxes);
101 -    I = insertObjectAnnotation(I,'rectangle',bboxes,scoers);
102 -    figure
103 -    imshow(I)
```

After training, this part can be tested by inputting test images. And can display the detection score on the bounding box (Refer to Figure 9).

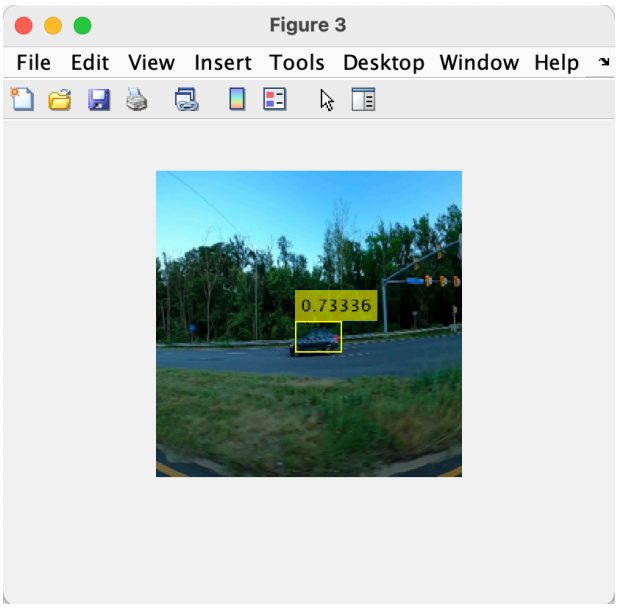


Figure 9# Object detection test results and detection scores

3. Description of software development process used

The object detection is used Matlab to develop, and all codes are running and implementing.

Firstly, it should follow in the feature extraction. using the Image Labeler which is in the Matlab to extract the features. It is essential to supply enough picture materials to extract features. So, in the Image Labeler, in order to extract the data of vehicles, it is provided a set of highway pictures. Marking each vehicle as an instance in each image. Then, export the vehicle ground truth to a workspace. Loading the vehicle ground truth data into Matlab, creating and combining the image and box label datastores. using it to train and validate the detector.

Secondly, establish a target detection network composed of two subnets. One is the feature extraction network, which is a pre-trained CNN. The other is to detect the network. The detection sub-network is a small CNN, which consists of several convolutional layers and YOLO v2 specific layers.

Then, the data augmentation is important to improve network accuracy. Using MatLab to randomly transform the original data, it can add more variety to the training data. Through randomly flipping the image and associated box labels horizontally, it will augment the training data. By adding more changes, it will improve accuracy of processing results. After that, it starts to train the object detector. Loading the pre-processed validation. Through using trainYOLOv2ObjectDetector (Computer Vision Toolbox) function to finish the training of object detector.

Lastly, evaluating the trained object detector on test set to measure the performance. measuring average precision to incorporates the ability of the detector to make correct classifications (precision) and the ability of the detector to find all relevant objects (recall).

Using detector to detect all the test images. Trying to improve the average precision, making the ideal precision is close to 1 at all recall levels. After the training and test, running the detector on an image, it will box select the vehicles in the image material, thus, realizing the subject detection of object detection.

4. Description of the evaluation methods

We chose to use a thousand data sets about cars. Originally, we planned to use 60% of the data set (600 pictures) for training, 10% of the data set for verification, and the rest for testing the trained detector. However, in the actual operation process, a considerable part of the images of the training set does not exist in the target that we need to detect, that is, the car, which undoubtedly has a great impact on the results of our training.

We use the object detector evaluation function of Computer Vision Toolbox™, and use the accuracy index to evaluate performance, The number provided by the average accuracy rate This number is a combination of the detector's ability to correctly classify (precision rate) and the detector's ability to find all relevant objects (recall rate). The function of drawing the PR curve is to intuitively show the pros and cons of performance, the larger the value, the stronger the performance

First, apply the same pre-processing transformation applied to the training data to the test data

```
preprocessedTestData = transform(testData,@(data)preprocessData(data,inputSize));
```

Then, run the detector on all test images.

```
detectionResults = detect(detector, preprocessedTestData);
```

Next, use the average accuracy index to evaluate the target detector.

```
[ap,recall,precision] = evaluateDetectionPrecision(detectionResults, preprocessedTestData);
```

Finally, draw the PR curve (accuracy rate/recall rate)

```
figure
plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f',ap))
```

After running, we can get the result (Refer to Figure 10).

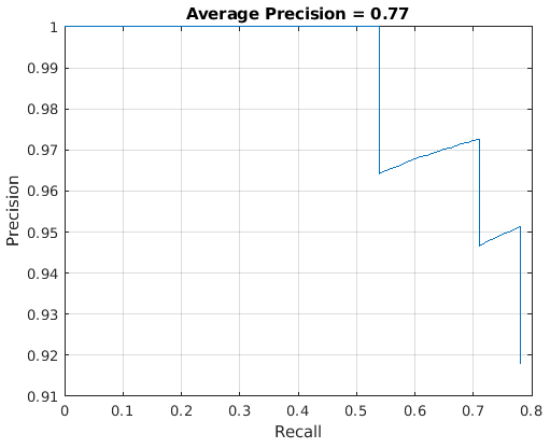


Figure 10# Average Precision

The value of IoU can also explain the accuracy of our design model to a certain extent. The value of IoU is the result of dividing the overlapping part of the two regions by the collective part of the two regions (Refer to Figure 11).

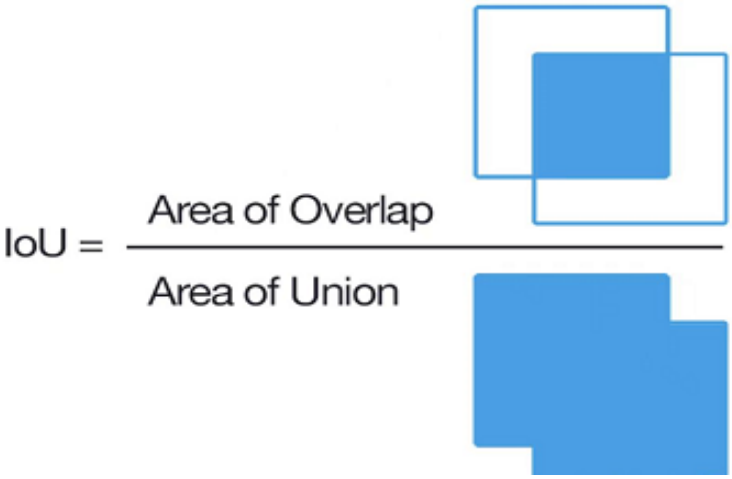


Figure 11# The calculation formula of IoU

The red square represents the accurate value, and the green square represents the predicted value, which means that the larger the value of IoU, the more accurate the model (Refer to Figure 12).

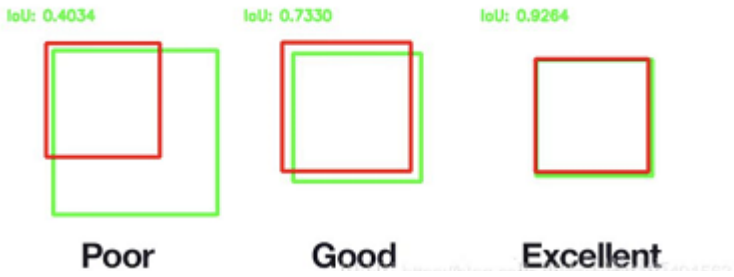


Figure 12# Different levels of IoU

The average IoU of our model is 0.7634, which is an acceptable result (Refer to Figure 13).

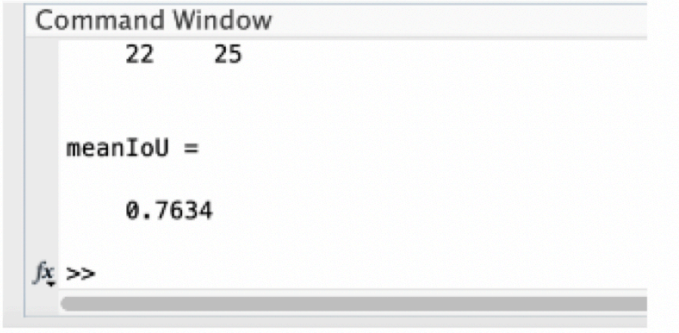


Figure 13# IoU training results of the training set

In addition, the value of RMSE is a description of the degree of dispersion, and does not represent the pairing error. It can be understood as an assessment of stability. The lower the value, the lower the degree of dispersion and the higher the stability. Its calculation formula is shown below (Refer to Figure 14).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figure 14# Calculation formula of RMSE

The RMSE value of our training project is 0.63513 (Refer to Figure 15). This shows that the degree of dispersion of the project is very low, and the stability of our project is perfect.

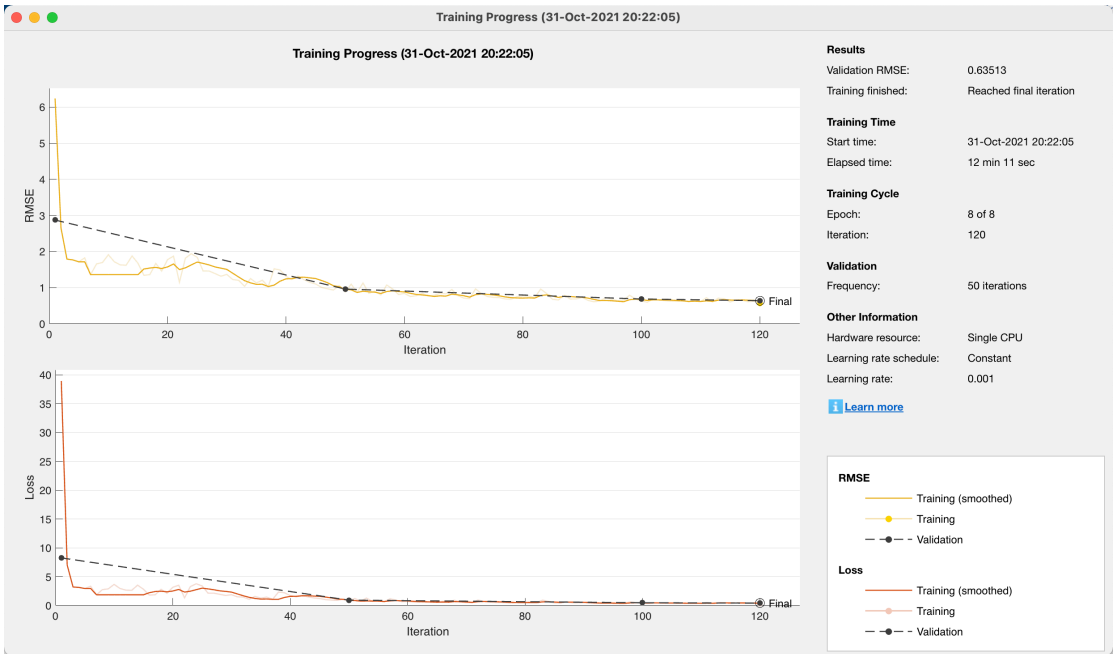


Figure 15# Training process

```
* vehicle

Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================|
|       1 |           1 |       00:00:23 |         6.24 |         2.87 |      38.9270 |       8.2483 |          0.0010 |
|       4 |          50 |       00:05:14 |         0.88 |         0.96 |       0.7761 |       0.9238 |          0.0010 |
|       7 |         100 |       00:09:59 |         0.63 |         0.69 |       0.4029 |       0.4711 |          0.0010 |
|       8 |         120 |       00:11:59 |         0.56 |         0.64 |       0.3109 |       0.4090 |          0.0010 |
|=========================================================================================================================|
Detector training complete.
*************************************************************************
```

Figure 16# Training result information

In general, the PR value and the average IoU value of our detector are both over 0.76, which is a relatively good result, and the RMSE data fully proves that the stable performance of the training program is very excellent. However, if we have more reliable data sets, our performance will be improved a lot.

5. Description of results and challenges

After we trained the Yolo V2 object detection network, we tested our resulting structure

several times through multiple training and validation data sets. We found some problems by looking at the data set. Firstly, we believe that the reasons affecting the accuracy of identification can be roughly divided into the following three types. The first problem is the speed of the vehicle, too fast speed leads to the target vehicle in the image is very fuzzy, thus reducing the accuracy of recognition. The second problem is the number of cars. When there are multiple vehicles in a picture, as shown in the upper right corner, there are three cars in the picture, but only two of them are recognized, and the other one is not. The last one is the weather. Too dark weather, such as late afternoon, when the light is dim, can also prevent vehicles from being correctly identified. As shown the image below(Refer to Figure 17), the image has multiple vehicles, but due to the dark weather, only one vehicle has been identified. Strangely, the vehicles identified in the image are black, while the white ones are not. Normally, white should be more recognizable than black, which is one of the odder things we found. We also found a problem, as shown in the image on the lower right, where there is only one vehicle in the image, but two vehicles are identified.

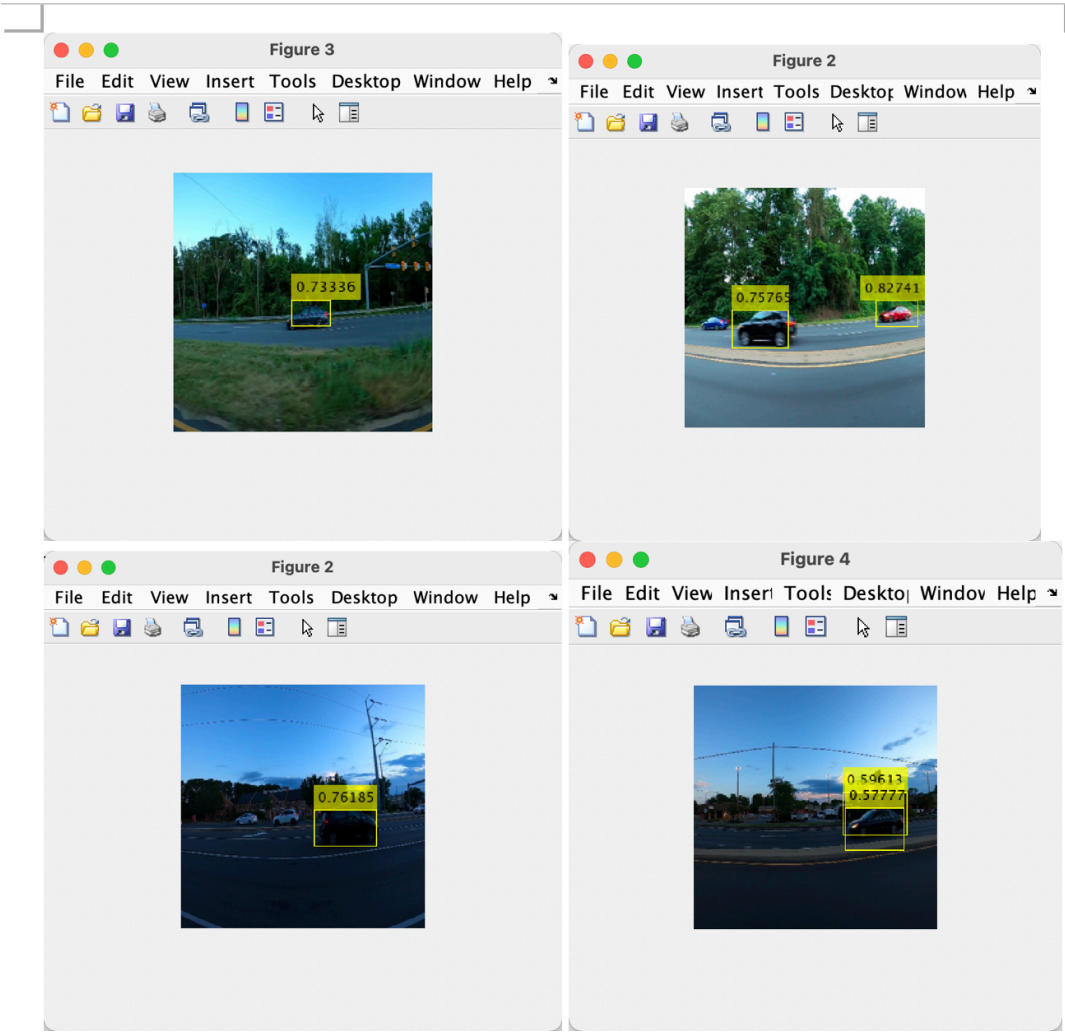


Figure 17# Vehicle detection results test

We think the problem is also related to the darker weather. By observing the above problems, we believe that this is the cause of the insufficient number of training sets. Of our 1,000 training sets, only 364 contained vehicles, and the remaining 636 contained no

vehicles. This results in a greatly reduced accuracy of vehicle recognition in the training set.

Secondly, we didn't have a lot of ideas about the code at first, but we found an example of building a YOLO V2 target detection network on the Math Works website. We analyzed and discussed the examples on the official website, and gradually had some ideas. However, during the testing process, we also encountered some problems with the code. When we use the Image Labeler to create labels, the path it takes to load the dataset is an absolute path, and when we test on another computer, the path is disabled. We have tried to solve this problem in the past, unfortunately due to time issues, this problem has not been solved, but we will continue to look at it in the future.

6. Description of the contributions of each team member

During the implementation of the project, we held regular remote meetings every week, and everyone maintained an active participation: discussing the progress of the project, the problems encountered and how we can solve them. Therefore, there is no doubt that our teamwork is pleasant. This not only allows us to learn about object detection technology together, but also allows us to learn how to communicate and solve problems when facing problems. Regarding the contribution of each member to the team, the details are as follows:

Question discussion	**All Members**
Code implementation	**Yangyang Jin**
Introduction to the problem	**Yangyang Jin**
High level description of the code	**Qibai Chen**
Description of software development process used	**Junting Song**
Description of the evaluation methods	**Tianyang Zhang**
Discussion of results and challenges	**Weilin Sun**
Project time control & Final integration	**Yangyang Jin**

7. Appendix containing a listing of the code with comments

```
load ass_1000_de.mat;

% Load dataset
imageFilename = gTruth.DataSource.Source;
vehicleDataset = table(imageFilename);
vehicleDataset = cat(2,vehicleDataset ,gTruth.LabelData);
vehicleDataset(1:4,:)

% Randomly shuffle the data set and
% divide the data set into training set, validation set and test set
rng(0);
shuffledIndices = randperm(height(vehicleDataset));
idx = floor(0.6 * length(shuffledIndices) );

trainingIdx = 1:idx;
trainingDataTbl = vehicleDataset(shuffledIndices(trainingIdx),:);

validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = vehicleDataset(shuffledIndices(validationIdx),:);

testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = vehicleDataset(shuffledIndices(testIdx),:);
```

```
% Convert the address to a picture
% Use "imageDatastore" to get the picture, "boxLabelDatastore" to get the bounding box label
imdsTrain = imageDatastore(trainingDataTbl{:,'imageFilename'});
bldsTrain = boxLabelDatastore(trainingDataTbl(:,'vehicle'));

imdsValidation = imageDatastore(validationDataTbl{:,'imageFilename'});
bldsValidation = boxLabelDatastore(validationDataTbl(:,'vehicle'));


% Combined image and bounding box label data storage
trainingData = combine(imdsTrain,bldsTrain);
validationData = combine(imdsValidation,bldsValidation);

% Display training images with bounding box
data = read(trainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,'Rectangle',bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)
```

```
% Create YOLO v2 Object detection network
% In order to reduce the computational cost of running the example, specify the network input size as [224 224 3],
% which is the minimum size required to run the network.
inputSize = [224 224 3];
numClasses = width(vehicleDataset)-1;

trainingDataForEstimation = transform(trainingData,@(data)preprocessData(data,inputSize));
numAnchors = 7;

% Display IoU score
% IoU is a measurement standard. This standard is used to measure the correlation between reality and prediction.
% The higher the correlation, the higher the value.
[anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingDataForEstimation, numAnchors)


featureLayer = 'activation_40_relu';
lgraph = yolov2Layers(inputSize,numClasses,anchorBoxes,resnet50,featureLayer);

% Data enhancement can improve the accuracy of the network by randomly
% transforming the original data during training.
% Use transform to enhance the training data by randomly
% flipping the image and the associated bounding box labels horizontally.
augmentedTrainingData = transform(trainingData,@augmentData);
```

```
71      % display the enhanced training data.
72 -    augmentedData = cell(4,1);
73 -    for k = 1:4
74 -        data = read(augmentedTrainingData);
75 -        augmentedData{k} = insertShape(data{1},'Rectangle',data{2});
76 -        reset(augmentedTrainingData);
77 -    end
78 -    figure
79 -    montage(augmentedData,'BorderSize',10)
80
81      % Preprocess the enhanced training data and validation data in preparation for training.
82 -    preprocessedTrainingData = transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
83 -    preprocessedValidationData = transform(validationData,@(data)preprocessData(data,inputSize));
```

```
85      % Read preprocessed training data
86 -    data = read(preprocessedTrainingData);
87 -    I = data{1};
88 -    bbox = data{2};
89 -    annotatedImage = insertShape(I,'Rectangle',bbox);
90 -    annotatedImage = imresize(annotatedImage,2);
91 -    figure
92 -    imshow(annotatedImage)
93
94 -    options = trainingOptions('sgdm', ...
95              'MiniBatchSize',16, ....
96              'InitialLearnRate',1e-3, ...
97              'MaxEpochs',8, ...
98              'CheckpointPath',tempdir, ...
99              'ValidationData',preprocessedValidationData,...
100             'Plots','training-progress');
101         %layer = detector.Network.Layers;
102 -     [detector,info] = trainYOLOv2ObjectDetector(preprocessedTrainingData,lgraph,options);
```

```
104     % tested by inputting test images. And can display the detection score on the bounding box
105 -   I = imread('vid_4_2280.jpg');
106 -   imshow(I);
107 -   I = imresize(I,inputSize(1:2));
108 -   [bboxes,scores] = detect(detector,I);
109 -   I = insertShape(I,'Rectangle',bboxes);
110 -   I = insertObjectAnnotation(I,'rectangle',bboxes,scoers);
111 -   figure
112 -   imshow(I)
113
```