POLITECNICO DI TORINO

Master's degree in ICT for Smart Societies

Master's Thesis

Lane Line Detection and Classification Based on Deep Learning

Supervisors
Prof. Tao Huang
Dr. Jie Zhao

Candidate
Yuhao Chen

Abstract

In the 21st century, with the progress of computing capability and the rapid development of machine learning, automatic driving technology is becoming more and more advanced. Nowadays, the field of autonomous driving technology has become a "strategic highland" for various vehicle companies. Lane line detection is a key task in this field, which plays a vital role in the decision-making of automatic driving. At present, deep learning has made great achievements in various fields of computer vision, and it is also widely used in the field of lane detection. Compared with traditional image processing method, the deep learning method is less affected by weather conditions and has higher generalization ability, so the lane line detected by this method are more stable. Current lane line detection algorithms still face some problems. For example, the detection is easily affected by illumination and occlusion, which will have a significant impact on the segmentation results. Besides, there was almost no literature that simultaneously detects and classifies lane lines. In this work, a two-stage end-to-end lane line detection-classification model is proposed. The main contributions of this work are as follows:

Lane line detection
The lightweighted semantic segmentation network ERFNet is used as the basic network to detect lane lines. The network has added a lane line existence prediction module to assist in predicting whether the lane line exists, and to guide the network to understand the differences among lane line instances.
Proposed a attention module called Balanced Attention Network to capture the interdependence of global features in both channel and spatial dimensions. While improving the segmentation performance of the model, there is no significant increase the number of parameters.

Lane line classification
Labeled on CULane dataset, the largest public lane line dataset. The lane line classification information was added on the basis of the original annota-

tions.
A lane line classification network is designed. After feature extraction and reconstruction of images, each lane line is classified through the classification network.

The proposed model is validated on the CULane benchmark and reaches 74.0 F1-Measure and The overall accuracy for lane line classification exceeds 90%. On the NVIDIA GeForce RTX 2080Ti GPU, the detection network and classification network can reach speeds of 24 ms and 7 ms, respectively.

Keywords: lane detection, lane classification, deep learning, computer vision

Acknowledgements

First of all, I would like to thank to my supervisor Prof. Tao Huang. During this work, he patiently listened to my stage report and proposed amendments in time to ensure the smooth progress of the thesis. Without his strictness and patience, this thesis would not have been possible.

I would also like to thank my co-director, Dr. Jie Zhao, the founder of Borden Intelligent Technology. He put forward many suggestions for improvement of my thesis.

I would also like to thank my colleagues from the company and school for their selfless help in this work.

The two-year master's study at the Politecnico di Torino is coming to an end. Finally, my thanks would go to my beloved family and girlfriend for providing me deeply support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Automotive electronics is developing towards automation, intelligence and network connection, and the autonomous driving industry has attracted increasingly more attention. With the development of various technologies, an increasing number of automobile manufacturers devote their energy to the research and development of self-driving cars. Among them, ADAS: Advanced Driver Assistance System is an essential functionality for achieving full autonomy. ADAS systems usually include automatic emergency braking (AEB), adaptive cruise control (ACC), parking assistance (PA), lane line and road detection, traffic signal and traffic sign recognition (RSR), and so on. Various sensors mounted on the vehicle (camera, laser radar, satellite navigation, millimeter-wave radar, etc.) can be used to precept the surrounding environment anytime and anywhere during the driving process of the vehicle. Meanwhile, through using those devices, the following processes can be conducted: data collection, real-time inference, and decision making as well as chassis control of vehicles. Thanks to them, driving comfort and safety of vehicle has been highly promoted.

In particular, lane detection is an essential and significant task. Recently, along with the development of semantic segmentation in deep learning, a batch of state-of-the-art algorithms have continuously improved lane detection technology. However, applying such techniques to vehicles still faces difficulties.

1.2 State-of-the-Art

Currently, there are two main methodologies in lane line detection: traditional one based on image processing and the new one relying on deep learning algorithms. This section will introduce the State-of-the-Art of these two methodologies.

1.2.1 The Traditional Methodologies

Lane line detection technology based on image processing has been widely applied to the auxiliary driving system of modern vehicles by features (color and shape) of lane lines. Figure 1.1 shows the main process of the first methodology.

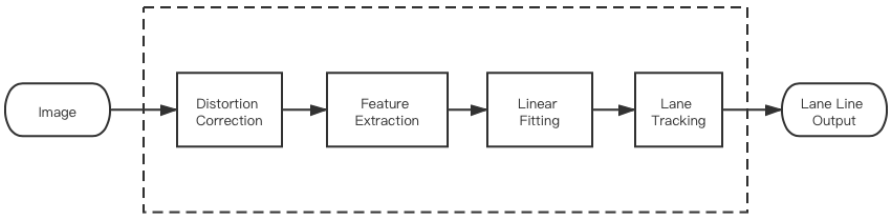


Figure 1.1: The traditional lane line detection process

Lane line detection algorithms based on traditional methods can be roughly divided into two categories, one is lane line feature-based approach, and the other is model-based approach.

Feature-based approach
In [10], the author used an inverse perspective transformation to convert the original image in the world coordinate system into a top view. According to the radical difference between the lane line and other targets on the road surface, an adaptive threshold is used to convert the top view image into a binary image. Finally, the lane line is extracted by Hough transform, and the lane line is fitted by the Random Sample Consensus(RANSAC) algorithm[11]. Kang et al.[12] proposed to use Sobel operator[13] for edge detection to obtain noisy lane line edge features. Then the road image was divided into multiple sub-regions along the vertical direction of the image, and dynamic programming was usedto extract the lane line . Collado et al.[14] proposed an adaptive lane detection and classification algorithm based on spatial lane features. The inverse perspective transform is used to convert the input

camera image into a perspective view of the road image, which removes the distortion of the camera's perspective.

Model-based approach
Model-based approaches usually assume that lane lines can be described by specific models, such as linear models, parabolic models, or other spline curves. In addition, most models are based on the assumption that the ground is flat. Zhou et al.[15] proposed a lane line detection algorithm based on geometric model and Gabor filter[16]. The inverse perspective transform is used to project the photo to the bird's-eye view, and the deviation caused by the angle of view is removed, so that the lane lines on the picture become parallel. Bosaghzadeh et al.[17] used principle component analysis (PCA) to solve the image perspective problem, and used a rotation matrix model to achieve lane line detection. Kim et al.[18] proposed to use HSV color space and the shape of lane lines to identify candidate lane lines, and then use Hough transform to determine the lane position.

Although the traditional methods has high processing speed, there are some limitations. For the simple driving environment in ideal weather, the traditional scheme can achieve high precision and recall rate. However, when the road condition is not ideal (such as traffic congestion, night traffic, lane lines are not obvious, etc.) or the weather is poor (such as rain, snow, fog, etc.), there is still shortages and imperfections on lane detection. In conclusion, the traditional method can generate reasonable results only in specific conditions, while it has visible disadvantages in other conditions.

1.2.2 The Deep Learning Methodologies

With the maturity of machine learning technology, neural networks have made amazing achievements in the field of image recognition. A surging number of people have begun to use deep learning to predict lane lines. Although many SOTA models have achieved high detection accuracy, most models need to consume more computational resources, which results in a long detection time. As low cost in calculation and high accuracy are trade-offs to each other, it is necessary to figure out a balance between them.

At present, the traditional image processing methods are unable to meet the ultra-high requirements of automatic driving for lane detection accuracy and speed. Compared with traditional methods, deep learning shows more advantages in the fields of computer vision. Seeing trends of development of deep learning, lane detection deserves a more competitive method to achieve

the expectation of autonomous driving system.

Although convolutional neural networks have powerful feature extraction capabilities. However, due to the slender shape of the lane line and the long distance in the space, convolutional neural networks occupy only a small pixel in the image, which can be easily affected. Pan et al.[1] proposed a new convolution method called Spatial CNN (SCNN), which slices and convolves network blocks layer by layer in four directions, so that spatial information is transmitted in different directions. In each direction, the current slice is passed to the next slice after convolution to extract features. After completing the feature transfer in one direction, switching the convolution block to the other direction and repeating the above steps. This method has a good segmentation effect on slender structures such as lane lines and telephone poles, and can solve the detection problem when lane lines are occluded.

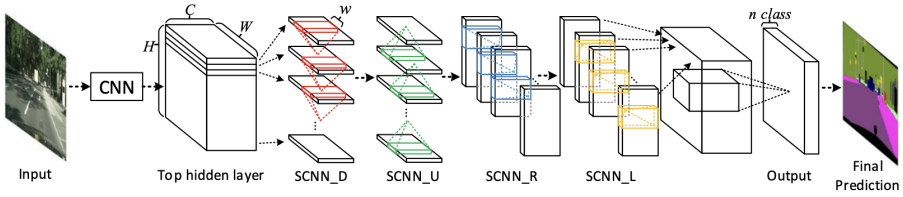


Figure 1.2: The network structure of SCNN[1]

In 2018, Neven et al. proposed an end-to-end lane line detection model LaneNet[2]. LaneNet consists of two branches: segmentation branch performs binary semantic segmentation on lane lines to distinguish whether pixels belong to lane lines or background, while embedding branch clusters the segmentation results and assigning pixels to different lane line instances. In addition, the author trained a neural network H-Net that can predict the perspective transformation matrix H. This method of predicting parameters through the network can adapt itself to terrain changes and improve the robustness of the model.

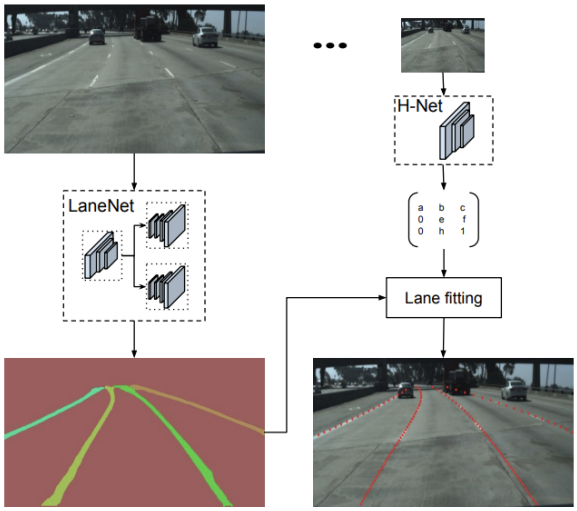


Figure 1.3: System overview of LaneNet[2]

In 2020, Lucas Tabelini et al. proposed PolyLaneNet[3], a new end-to-end lane line detection model. PolyLaneNet uses polynomials to describe lane lines. After extracting image features through the convolutional network, the model passes through a fully connected layer, and finally outputs the confidence of each lane line and the relevant third-order curve coefficients.

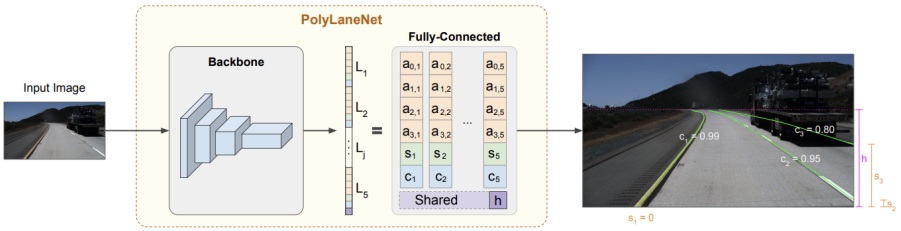


Figure 1.4: System overview of PolyLaneNet[3]

Fabio Pizzati et al.[19] proposed to use two cascaded networks to detect and classify lane lines. The first network outputs the segmentation results of the lane line. According to the segmentation results, the lane lines are sampled on the original image, and then the extracted pixels are put into a designed descriptor in order. Finally, the descriptor is input into the second network to classify the lane lines.

1.3 Motivation and Task Description

1.3.1 Motivation and Objective

Lane line detection and classification are challenging tasks. The lane lines of urban roads vary widely, and the road conditions are complex. Figure 1.5 shows several scenes where the traffic situations are complex for lane line detection. The followings are some difficulties of current lane line detection tasks.

1) The lane lines occupy a small proportion in the image. Compared to other targets, the shape of lane lines is slender and variable.

2) Lane lines are easily obstructed. When the lane line is sheltered from other vehicles, it brings difficulties to detection.

3) The color and the shape of the lane line are similar to other lines and signs on the ground.

4) The shape of the lane line is variable and the color differs from each other. In various scenes (such as straights, curves, crossroads, etc.), different lane lines are difficult to recognize.

5) In practical applications, in order to meet the needs of autonomous driving, real-time performance is the most important factor for detection tasks, resulting in a low computing requirement during inference.

Figure 1.5: Complex traffic scenarios

For autonomous driving projects, the task of lane line detection is not only to accurately detect the lane line itself, but also to be able to distinguish the type of lane line and the location of the lane line. This information is necessary for subsequent trajectory planning. There is almost no literature that simultaneously detects and classifies lane lines. with this in mind, our objective is to build a lane line detection-classification model that can detect

the lane line, the position of the lane line, and also can distinguish the types of lane lines.

After understanding the characteristics of lane lines and the difficulty of lane line detection tasks, the advantages and disadvantages of traditional methodologies and deep learning methodologies were compared. As the conclusion, the deep learning method was decided to carry out the task of lane line detection.

In this work, we propose a two-stage end-to-end lane line detection-classification model. In the first stage, the network uses semantic segmentation to detect lane lines. The lane lines are extracted from the image according to the segmentation results. Then input the extracted lane line image into the second stage network for classification.

(a) Image (b) Detection (c) Classification

Figure 1.6: Visualization of lane Line detection and classification

1.3.2 Contents Overview

This thesis consists of 5 chapters in total, this section will give a brief summary of these five chapters:

Chapter 1: Introduction
This chapter mainly introduces the research background of lane line detection. The State-of-the-Art of traditional methodologies and deep learning methodologies were reviewed. Finally, some difficulties in current lane line detection are listed.

Chapter 2: Data and Data Augmentation
This chapter offers the overview of the dataset used in this work, as well as the production process of the lane line classification dataset. To further improve the generalization performance of the model for various scenario detection, we expanded the amount of data. In this section, we describe the data augmentation operations we used in detail.

Chapter 3: Lane Line Detection-Classification Model
At the beginning of this chapter, we introduce the related deep learning techniques used in this work. First, the basic structure of convolutional neural network techniques is introduced. Then two important semantic segmentation networks are described in detail. Finally, the application of Attention mechanism in computer vision is introduced.

Then, we present a detailed description of the lane line detection-classification model proposed in this work. Firstly, the overall structure of the model is introduced. Then the semantic segmentation module, the Attention module, the lane line presence prediction module and the lane line classification module of the network are described in detail. Finally, the loss function used in this experiment is introduced.

Chapter 4: Experiments and Results
The environment of the experiment is introduced. Then the data preprocessing, the setting of training parameters and the process of extracting lane line features are described. Then the evaluation metrics and evaluation steps are introduced. In the results part, a series of comparative experiments are carried out to verify and evaluate the performance of the improved part proposed by this work.

Chapter 5: Conclusions and Future Work
This chapter concludes the lane line detection-classification model proposed in this work and provides suggestions for future work.

Chapter 2

Data and Data Augmentation

2.1 Dataset Overview

In this project, we choose CULane[20] as our datasets. CULane datasets is a large-scale lane line detection datasets published by the Multimedia Lab at Chinese University of Hong Kong. All the video data was taken from the dashcams of six taxis in Beijing over a total of 55 hours. The entire datasets captured 133235 frames of images from the recorded video. Among them, 88880 frames are the training set, 34680 frames are the test set, and 9675 frames are the validation set. Compared to the TuSimple datasets[21], which is also widely used, the CULane dataset's scene is much richer, with diverse urban, rural and high-speed scenarios, more data, and more complex road conditions.

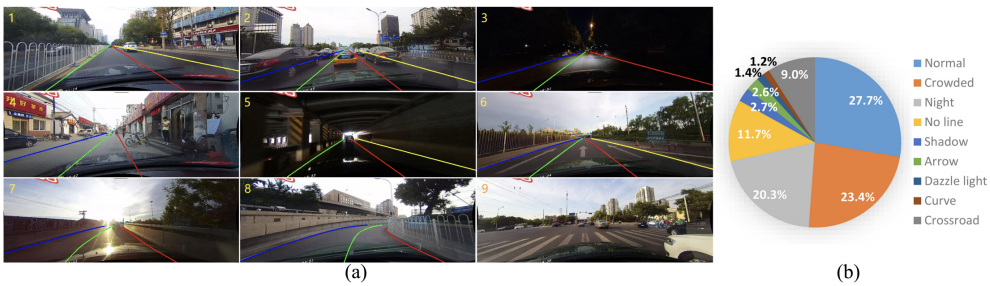


Figure 2.1: Scenarios distribution of CULane datasets: a)Dataset examples for different scenarios. b)Proportion of each scenario.[1]

Table 2.1: The image distribution of CULane test dataset in each scene

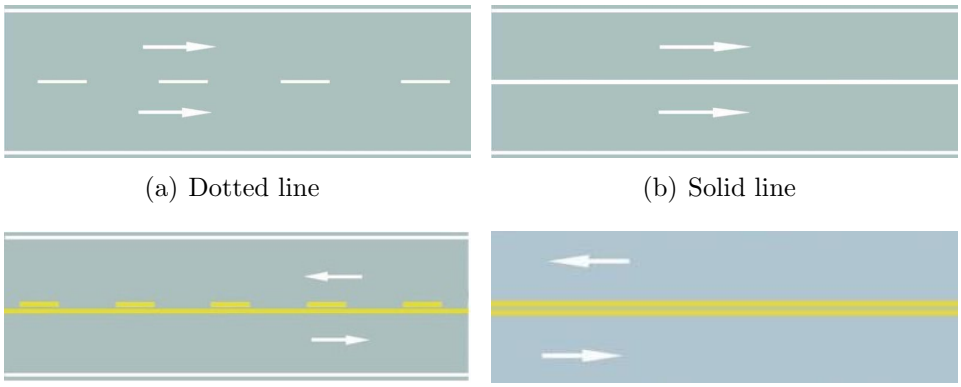
Category	*Description*	*# Images*
Normal	Lane lines are clear and there are fewer vehicles on the road.	9621
Crowded	There are more obstructed parts of the lane line; more vehicles on the road.	8113
Night	The overall ambient light is low.	7029
No-line	Lane line is not obvious or does not exist.	4067
Shadow	Shadows of other objects are projected on the lane line.	930
Arrow	Arrow signs on the road.	890
Dazzle light	Brighter light.	486
Curve	Curved road line.	422
Crossroad	At an intersection, there is no lane line.	3122
Total		34680

Each image is labeled up to four lane lines: the left lane line of the current driving lane, and the outer lane line of the adjacent lane, but not the opposite lane line. Besides, if the lane line is blocked (such as traffic jams, etc.), the lane line is also marked artificially, which is conducive to improving the robustness of detection.

2.2 Data Annotation

On the basis of the original CULane dataset, we marked the category of lane lines in each picture. After collecting statistics on the types of lanes on Chinese roads, we divide the lanes into 3 categories in our dataset: ***dotted line***, ***solid line***, and ***undefined line***.

It should be pointed out that for the dotted-solid line, if solid side of the dotted-solid line is close to the current traffic lane of the data collection vehicle, the dotted-solid line is considered as a solid line. Otherwise, it is marked as a dotted line.



(a) Dotted line (b) Solid line

(c) Dotted-solid line (d) Double solid line

Figure 2.2: Common types of lane line in China

All labeling work is done on the Boden Annotation Web platform(BAW) developed by Boden Intelligent Technology Co., Ltd[22] Figure 2.3. According to the type of lane line directly in front of the vehicle at the current position, a label is given to each lane line.

Since the images in the CULane dataset are obtained from video clips, there are many repeated scenes. We selected a total of 10060 images to label the types of the lane lines. Among them, 9060 for training and 1000 for validation.

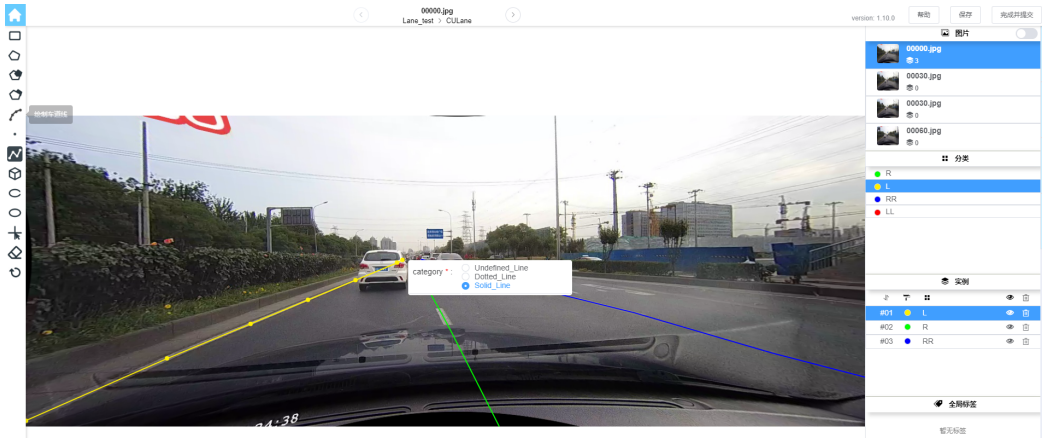


Figure 2.3: Boden Annotation Web platform(BAW)

Figure 2.4: An annotation example showing the attributes of the left lane line

After labeling is completed, export the result. Each image has a corresponding json file containing the relevant labeling information. Figure 2.4 shows

an example.

In the subsequent training and inference of the model, we need to create a mapping between label information and computer language. When generating the label of the lane line category, we will use a number to indicate the corresponding lane line category. The category is shown in Table 2.2.

Category	**Number**
Not Exist	0
Dotted Line	1
Solid Line	2
Undefined Line	3

Table 2.2: Lane line category label

2.3 Data Augmentation

Deep learning usually requires a large amount of training data to get more desirable results. In the case of limited amount of data, the amount of data can be expanded by data augmentation to increase the diversity of training samples to reduce the reliance of the model on certain attributes. At the same time, data augmentation can improve the generalization ability of the model to avoid overfitting.

Number of video clips	763
Total frames	88000
Max number of frames	180
Min number of frames	8
Average number of frames	115

Table 2.3: Information of training set

From Table 2.3 we can see that the CULane training set has 763 video clips with 88000 frames. Each clip contains 115 frames on average, so most of the data scenes are duplicated. In order to improve the generalization ability of the model, the data needs to be augmented.

Crop and Resize

For lane line detection tasks, real-time is an important factor that must be considered. Reducing the size of the input image can improve the inference speed of the model. Before the image is input to the model, the image needs to be cropped and resized. it is known by common sense that the lane lines do not appear in the sky, so that the sky part is first cropped before the image is input to the model, about 240 pixel points in height. And then resized the cropped image to 208x976.

Random Rotation

Curve detection has been one of the difficulties in lane line detection. The main reason for the low accuracy of segmentation for curved scenes is that most lane lines are straight, and the amount of data for curved scenes is relatively low.In the CULane datasets, the curved lane scenes only account for 1.2% of all training sets. During the training process, the model is unable to learn the characteristics of curved lanes, and instead, it is more likely to

ignore this part of data as noise. In order to increase the amount of data for the curved scenes, a random rotation data augmentation method was used in the preprocessing stage. The rotation range is from -30 to +30 degrees with a probability of 0.5. In this way, it is possible to increase the amount of data of the curved scenes by 50%, thus improving the performance of the model in segmenting the curved scenes.

Random Lane Erasing

Traffic jam is a common scenario in lane line detection task. Therefore, there are scenarios where the lane lines are partially or completely blocked by other vehicles. In this case, it is extremely important to accurately predict the location of the blocked lane lines. In this work, a data augmentation method is called random lane erasing designed: by cropping some areas in the original image, we simulate the scene of lane blocking when the road is congested. This method can improve the robustness of the model for lane line detection.

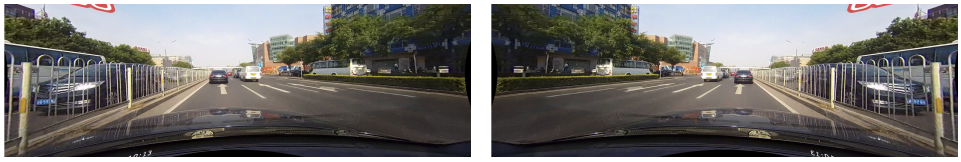
(a) Origin Image (b) Random Lane Erasing

Figure 2.5: Example of Random Lane Erasing

Figure 2.5 shows an example of Random Lane Erasing augmentation. The position of each erase block is randomly generated in the image, while the edge length of each erase block is between 50 and 100 pixels. Up to 3 erasure blocks will appear in each image. It should be noted that Random Lane Erasing only works on the original image and does not perform this operation on the ground truth.

Random Horizontal Flip

Flip is the most commonly used method in data augmentation. By flipping the image in the horizontal or vertical direction, the positional relationship between the pixels in the original image would be changed. Since lane lines will only be on the ground and not in the air, the data can be augmented by horizontal flipping to increase the diversity of the images.



(a) Origin Image (b) Random Horizontal Flip

Figure 2.6: Example of Random Horizontal Flip

Figure 2.6 shows the example of random horizontal flip. After the picture is flipped, the position of the lane line also changed. We can see that the solid line originally on the left is flipped to the right of the image and the dotted line that was originally on the right came to the left of the image. While flipping the original image, we also need to flip the ground truth and change the value of the lane line pixels to reorder the position of the lane lines.

With the above several data augmentation operations, the amount of data is effectively expanded, especially for lane lines in curved scenes and obscured scenes. As a result, the generalization ability of the model in different scenarios and the robustness of the model are further improved.

Chapter 3

Lane Line Detection-Classification Model

3.1 Convolutional Neural Network

Convolutional Neural Network (CNN) has been widely studied in various computer vision tasks because of its powerful feature extraction capability and excellent performance in experiments.

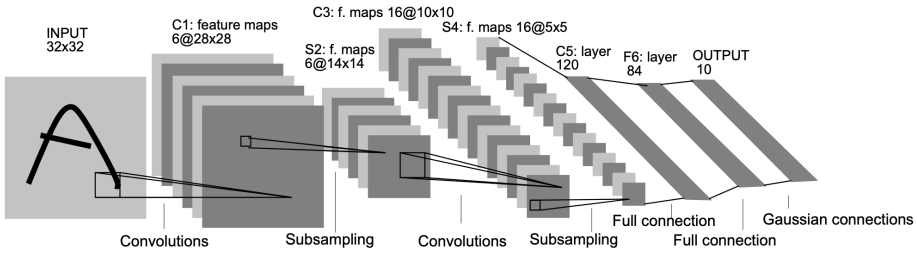


Figure 3.1: Architecture of LeNet-5[4]

Figure 3.1 is the structure of the classic image classification network LeNet-5, which clearly shows the basic structure of Convolutional Neural Network:

1) ***Convolution layer*** As a core part of CNN, the convolutional is responsible for extracting image features. By inputting the feature map to the convolutional layer, the convolution kernel will slide on the convolutional layer, and the distance of each sliding is called strip. The convolution kernel and the element at the current coverage position are weighted and summed at the corresponding position to obtain the value of the element

on the new feature map. Figure 3.2 shows the process of convolution operation.

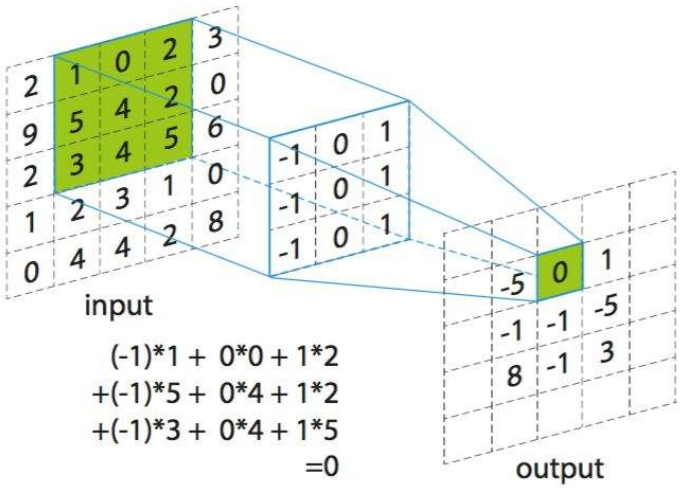


Figure 3.2: The 5x5 feature map passes the 3x3 convolution kernel[5]

2) ***Pooling layer*** Pooling can further reduce the image size obtained by convolution without changing the depth of the channel. The pooling can convert a higher-resolution picture into a lower-resolution picture, reducing the number of parameters of the network.

Common pooling operations include max pooling and average pooling. When doing the maximum pooling operation, a spatial neighborhood was designed and took the largest element from this spatial neighborhood as the output. And average pooling was to output the average of all elements in that spatial neighborhood. The pooling operation only targets the two dimensions of height and width, without changing the channel of the feature map.

Figure 3.3 shows the pooling operation with 2x2 kernel size and strip is 2.

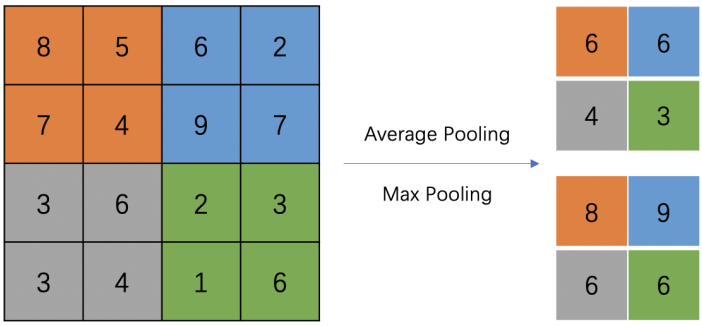


Figure 3.3: Pooling

3) ***Fully connected layer*** While the convolutional layer and pooling layer implement the feature extraction function, the fully connected layer maps the learned features to the label space of the input data. In the recognition task, the fully connected layer functions as a classifier.

Convolution layers extract the local features, and the pooling layer is used to decrease the network size while preventing overfitting. The fully connected layer integrates the extracted features. Finally, the classification result is obtained through the softmax layer.

Convolutional neural networks mainly have the following characteristics: local connection, weight sharing and multi-core convolution.

- ***Local Connectivity*** The receptive field is defined as the region in the input space that a particular CNN's feature is looking at[23]. In the matrix data of the image, the value of each position is usually highly correlated with the surrounding values: the closer pixels are more correlated, the features are more similar, and the probability of belonging to the same object is greater; the farther pixels are correlated weaker. The correlation between local pixels in the image is relatively high, and the characteristics of the image can be obtained by strengthening the connection between the local pixels.

- ***Weight Sharing*** Weight sharing means that each position on the feature map shares a convolution kernel weight instead of having all the independent weights for each position. By sharing weights, the network size could be further reduced.

- ***Multi-kernel Convolution*** For extracting the features of the picture, multiple convolution kernels are used for computing convolution

on the image, where the number of extracted features is the same as the number of convolution kernels.

3.2 Dilated Convolution

In general, semantic segmentation, we usually use convolutional layers and pooling layers that in semantic segmentation were used to increase the number of channels to adjust the area of the receptive field. Each convolution process will decrease the resolution of the feature map. In the process of upsampling, there will be a loss of accuracy. In order to reduce this loss of precision, the dilated convolution[24] has emerged. The dilated convolution can keep the resolution of the features unchanged while the receptive field is increased, thereby replacing the downsampling and upsampling operations. The information obtained on a large scale is very important in computer vision tasks, especially for semantic segmentation. Introducing a larger receptive field can improve the accuracy of segmentation.

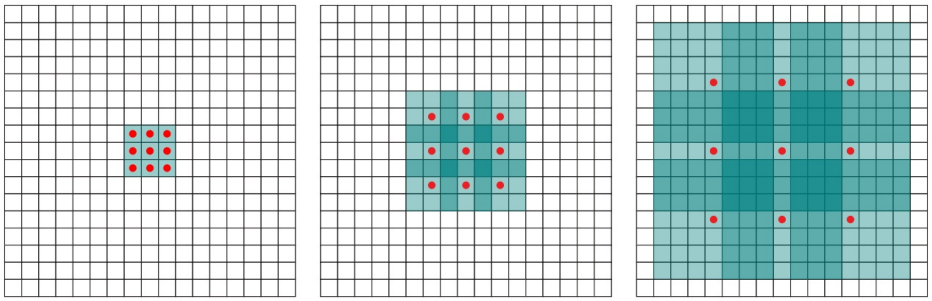


Figure 3.4: Dilated Convolution[6]

Since the kernel size of the dilated convolution is the same as the ordinary convolution, the size of the neural network remains unchanged. The dilated convolution has a larger receptive field than the ordinary convolution, which is adjusted by a dilation rate parameter. When different dilation rates are set, the receptive fields will be different, thus obtaining multi-scale information. The calculation formula for the output resolution of the feature map based on dilated convolution is as follows:

$$kernel_{dilated} = (DilationRate - 1) * (kernel - 1) + kernel \quad \text{(Eq.3.1)}$$

$$Feature_{out} = (Feature_{in} - kernel_{dilated} + 2 * padding)/stride + 1 \quad \text{(Eq.3.2)}$$

3.3 Semantic Segmentation

Semantic segmentation is another important research direction in computer vision fields. A pixel-wise classification is conducted in semantic segmentation, while image-wise classification is performed in image recognition tasks. This section will introduce two remarkable semantic segmentation models: Fully Convolution Networks and SegNet.

Fully Convolution Network
Fully Convolution Network(FCN) [25] is a representative work of deep learning application in the image segmentation field. And it has become the basic framework of semantic segmentation.

For a general CNN network, the last few layers are composed of fully connected layers, and then the category information of the image is obtained through Softmax. However, this one-dimensional result is the classification information of the entire image, not the classification information of each pixel. FCN proposes to replace the last few fully connected layers with the convolutional layers. Then generate a two-dimensional feature map, and output a segmentation map with the same size as the original image after upsampling. However, FCN also has obvious shortcomings:

- Since the upsampling process is a simple deconvolution, the result obtained is not fine, and the accuracy of image segmentation is not high enough.
- Did not consider the relationship between pixels, lack of spatial consistency.

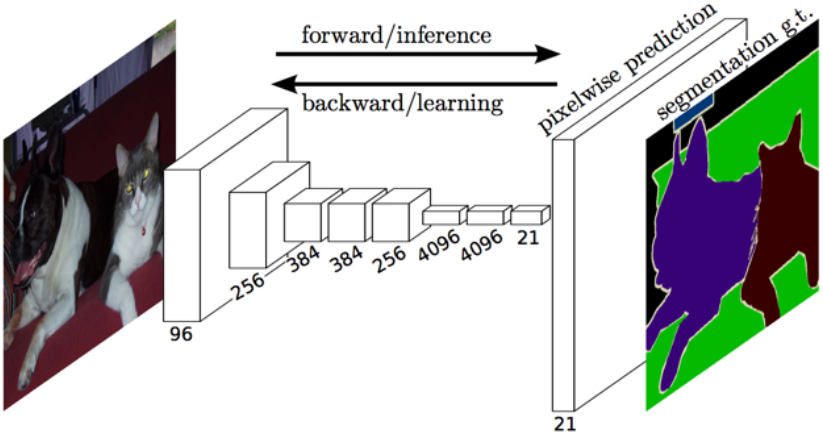


Figure 3.5: Fully Connection Network

SegNet

The SegNet[7] proposed by Vijay et al. in 2015 provides an Encoder-Decoder idea for the semantic segmentation network.

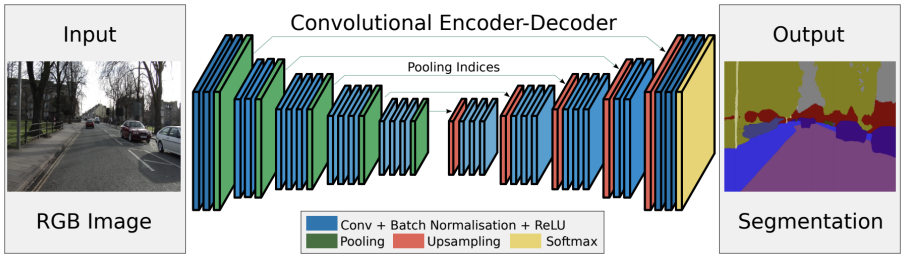


Figure 3.6: The Network structure of SegNet[7]

As shown in Figure 3.6, the network is designed as a symmetrical structure. The Encoder part extracts features of the input image, while the Decoder is used to restore the image detail information lost due to pooling during the Encoder process. Compared with the skip architecture of FCN, the recovery of spatial information in the Encoder-Decoder structure in SegNet is a process of multiple gradual recoveries. In this way, multiple scales of shallow semantic information can be combined to make the restoration of spatial information more accurate. The Encoder-Decoder structure is widely used in the field of semantic segmentation.

3.4 Attention Mechanism

Nowadays, inspired by a promising success in natural language processing tasks, the attention mechanism concept has been increasingly studied and adopted in diverse applications, including natural language processing and computer vision. The attention mechanism can be autonomously learned by the neural network, turning to help the public understand the world that the neural network sees. The so-called Attention mechanism is a mechanism that focuses on local information, such as a certain area in an image. As the task changes, the area of attention is also changing. In the task of road segmentation, an attention mechanism is often necessary to distinguish some confusing targets, such as roads and sidewalks. The attention mechanism helps the network focus on more critical features and suppress unnecessary features, improving the ability of the model to segment objects.

SENet
SENet[8] is a typical channel attention mechanism network, and proposes the Squeeze-and-Excitation Module(Figure 3.7). Channel attention can be understood as what the neural network is looking at. Each layer of the CNN has many convolution kernels, and each convolution kernel corresponds to a characteristic channel. Score the features generated by each channel, and suppress irrelevant features and enhance useful features according to the corresponding score.

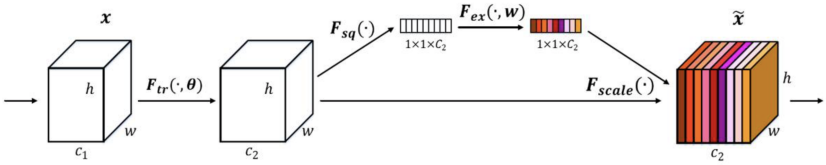


Figure 3.7: Squeeze-and-Excitation Module[8]

- ***Squeeze operation***: Apply a Average Pooling to the feature map X with C channels and output a 1x1xC feature map.

$$F_{sq}(x_c) = \frac{1}{H \times W} \sum_{i=1}^{W} \sum_{j=1}^{H} x_c(i,j), F_{sq}(x_c) \in R^C \qquad \text{(Eq.3.3)}$$

- ***Excitation operation***: After two fully connected layers, the sigmoid activation function is applied to limit the range of values between 0

and 1. Multiply this value as the scale to the C channels of X and use it as the input data of the next stage.

DANet

The self-attention mechanism is an improvement of attention mechanism. It selects more significant information for the target task from the global information. Therefore, all the feature information of the image can be used well. At the same time, the self-attention makes the network be concentrated on itself without other additional information so as to better deal with the dependence of long-distance or multi-level information in the image. Figure 3.8 is a network diagram of the self-attention mechanism.

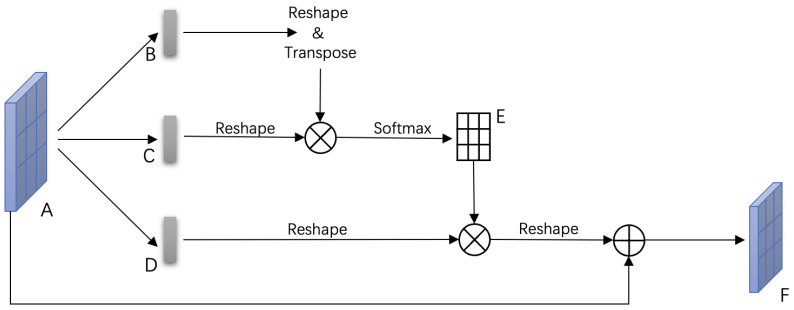


Figure 3.8: Architecture of Self-Attention network

DANet[9] is a classic application of the self-attention mechanism. DANet is a classic application of the self-attention mechanism, where the spatial and channel feature dependencies are captured separately:

- **Spatial dimension:**. The spatial attention module tells the network to look where. Spatial attention adjusts the attention of each position of the feature map, so that the model pays more attention to the area that deserves more attention.
- **Channel dimension:** The channel attention module tells the network to look what. First calculate the relationship between all channels on the feature map. Then a weighted summation is used for all channels to aggregate the channel characteristics and redistribute the weight of each channel.

The network architecture of DANet is shown in Figure 3.9. First, the output dimensions of the two attention modules are converted through the convolutional layer, and an element-wise summation is performed to achieve feature

fusion. Finally, convolution is followed to obtain the final predicted feature map.

The DANet establishes interdependency between features to obtain global contextual information, which helps to obtain more accurate segmentation results.

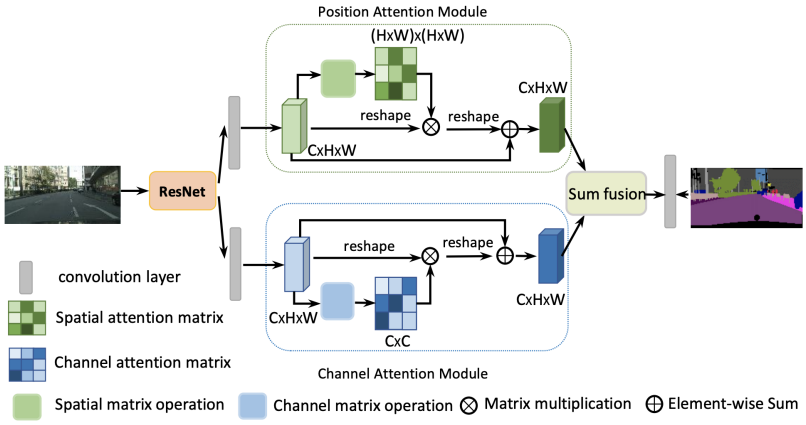


Figure 3.9: DANet[9]

3.5 Model Overview

For the semantic segmentation part, we chose ERFNet[26] as the backbone for feature extraction. ERFNet is a lightweight real-time semantic segmentation network. Since the shape of the lane line is slender and the number of pixels only occupies a small part of the whole picture, we add an improved attention module between the encoder and the decoder. Since this is the highest level of the network feature extraction part, the receptive field of each point is large, the semantic information is also richer, and more meaningful semantic representations can be obtained.

After extracting the features by an encoder, a lane line prediction model is layered to detect if all lane lines exist or not. Through this module, the network can be guided to acquire the global information of the entire picture.

After the decoder outputs the segmentation map of the lane line, we combine the output segmentation map with the input original image for feature extraction, and pass it through a lane line classification network to classify the type of each lane line.

Finally, the outputs of these three modules are merged to get the final result. The overall architecture of the lane line detection-classification model is shown in Figure 3.10:

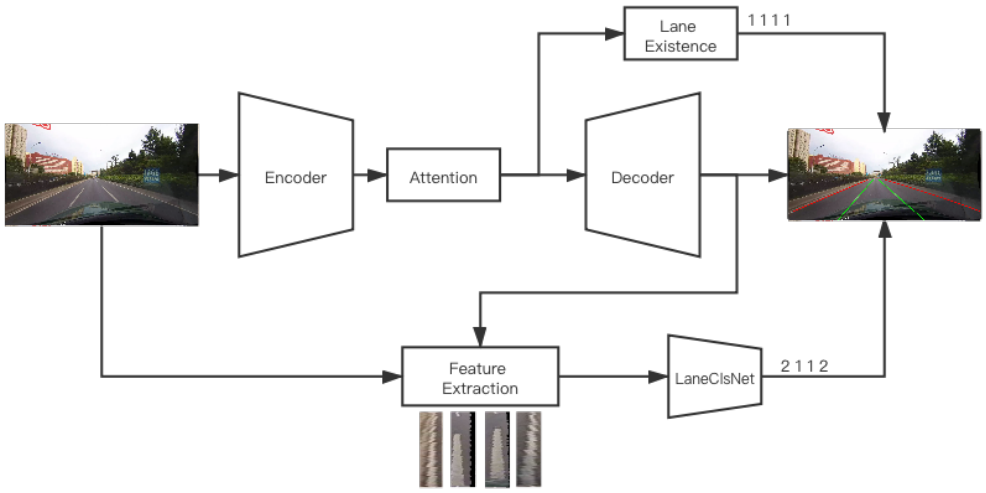


Figure 3.10: System architecture of the inference model

3.6 Semantic Segmentation Module

In this work, a lightweight semantic segmentation network ERFNet[26] is selected as the backbone after comprehensive weighting the segmentation precision, inference time and model size. Thus, high efficiency and accuracy of real time segmentation can be achieved by only few layers.

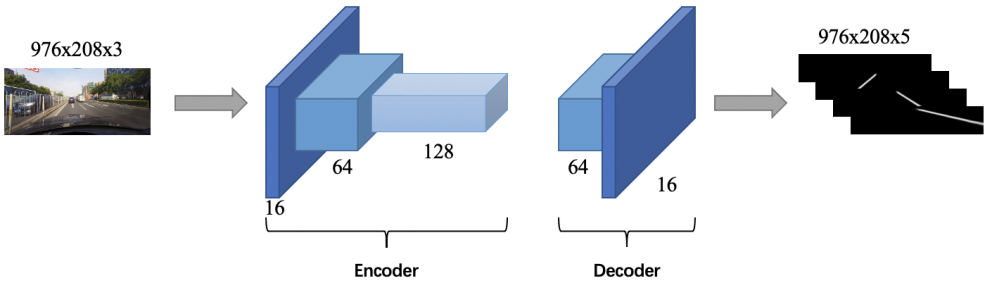


Figure 3.11: Architecture of ERFNet

Figure 3.11 depicts the network architecture of ERFNet, which adopts end-to-end Encoder-Decoder structure. The model has 23 layers, of which, layers 1-16 are the encoder stage, and layers 17-23 are the decoder stage. The Encoder part is similar to the conventional classification network. After each downsampling layer, the number of channels increase as the image resolution lowers. Finally, output the 128-dimensional feature map. For the Decoder part, upsamples these feature maps to the original resolution, and the output is the segmentation results of the same size as the original picture. Because of the use of hierarchical upsampling, there is no need to use skip joins to optimize the output as with FCN.

In the encoding stage, the Non-BN-1D module[26] is used to extract features. The Non-BN-1D module is a convolution with a 1-dimensional decomposition kernel constructed using a non-bottleneck structure. Non-bottleneck-1D decomposes the two 3×3 convolution kernels in the conventional residual module into two sets of 3×1 and 1×3 one-dimensional convolutions. Using the residual connection to fuse the input feature and the convolved feature map, which strengthens the expressive ability of the network. Some specific layers in the Non-BN-1D module utilize dilated convolution with expansion rates of 2, 4, 8, and 16, respectively, in order to expand the receptive field without increasing the amount of parameters, resulting in an obvious improvement on detection effect. The comparison between Non-BN-1D module and residual module is shown in Figure 3.12.

(a) Residual module (b) Non-BN-1D

Figure 3.12: Comparison between residual module and Non-BN-1D module

3.7 Attention Module

The attention module used in this work refers to the SENet and DANet structures, and proposes a Balanced Attention Network (BANet) that combines channel-dimension attention and spatial-dimension attention.

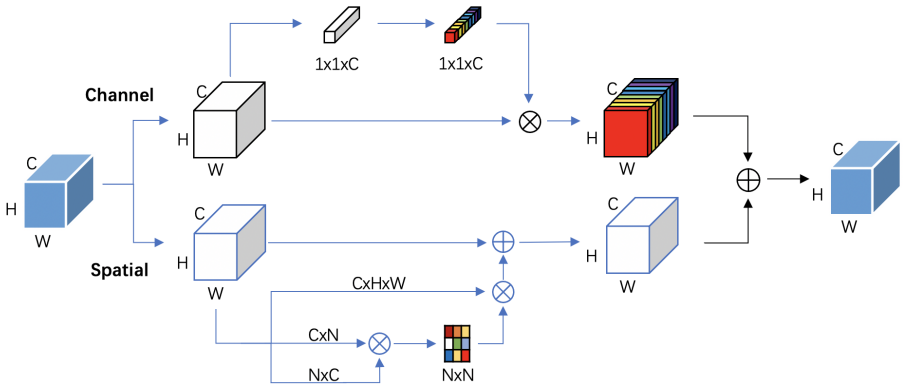


Figure 3.13: Architecture of BANet

Figure 3.13 shows the architecture of BANet. In the channel attention part, the Squeeze and Excitation operations of SEnet are used. BANet integrates and compresses channel information by applying pooling in the channel dimension for each point in the space, and finally redistributed the channel weights.

In the spatial attention part, for an input feature map SA, $SA \in R^{C \times H \times W}$. First of all, pass SA through a convolutional layer, a BatchNorm layer

and a Relu activation to generate three new feature maps Q, K, V, where $Q = K = V = SA \in R^{C \times H \times W}$. Reshape Q, K into $Q \in R^{C \times N}$, $K \in R^{N \times C}$, where $N = H \times W$. Then multiply the two matrices of Q and K, and get the final spatial attention map with shape $R^{N \times N}$ through a Softmax layer. Then do matrix multiplication of the spatial attention map and V, and transpose it back to shape $R^{C \times H \times W}$. Finally, the outputs of the two attention modules are added to obtain a further enhanced feature expression.

The attention module can be added to any position of the network. In this work, after experimental comparison, we found that the segmentation performance is best when BANet is placed after the last downsampling layer. As it contains the richest feature information at that position.

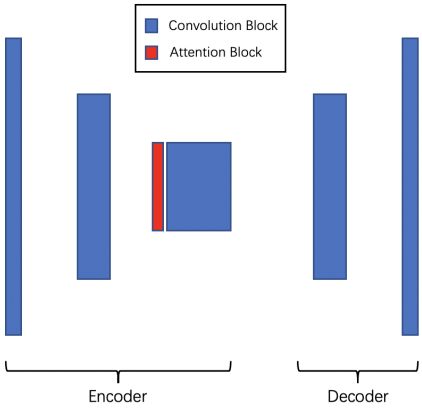


Figure 3.14: The position of the Attention module in the network

3.8 Lane Line Existence Prediction Module

For the semantic segmentation task of lane lines, the CULane dataset used in this project has no more than four lane lines labeled on each picture. Therefore, a branch is added after the encoder extracts the features in order to predict whether the lane line exists.

The semantic segmentation network essentially performs category prediction for each pixel. Furthermore, global information is not sufficient to perform the prediction task. It is more difficult to understand contextual information without global information. By adding the lane line existence prediction branch after the encoder extracts the features, the network can be guided to understand the differences among lane line instances, and can assist in predicting whether the lane line exists.

The structure of the lane line existence prediction module is shown in the Figure 3.15. From the output of the feature map from the encoder, the features are further extracted through two consecutive convolution modules. The Softmax was used to normalize the output feature map in the channel dimension. In the end, two continuous fully connected layers and the sigmoid activation are used to obtain a vector with a dimension of 1x4. Each value represents the probability that the network predicts the existence of each lane line.

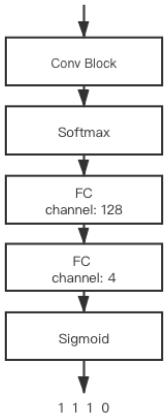
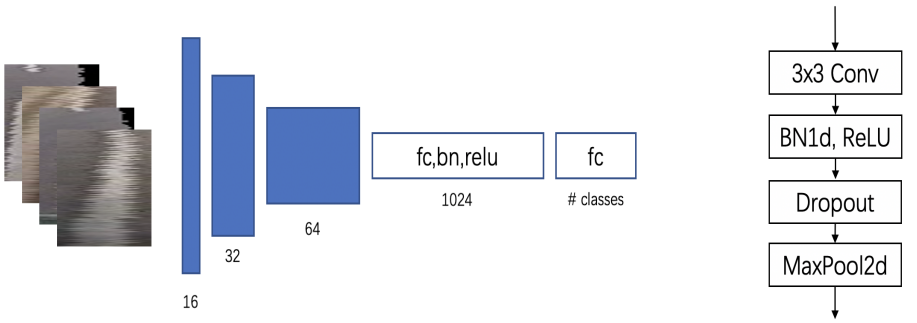


Figure 3.15: Lane Existence Module. For the output result: 1 means that the lane line exists, and 0 means that it does not exist.

3.9 Lane Classification Module

For autonomous driving, it is insufficient to only detect the position of the lane lines. The type of lane line must also be taken into consideration. Only by providing correct lane line position information and category information, the subsequent trajectory planning can make correct decisions.

In this work, we designed a lane line classification network to classify the lane line images after feature extraction.



(a) LaneClsNet (b) ConvBlock

Figure 3.16: Architecture of LaneClsNet

The structure diagram of lane line classification network is shown in Figure 3.16, which is mainly composed of three successive convolutional blocks(ConvBlock) and two fully connected layers. The final lane line is divided into 3 classes: dotted line, solid line and undefined line.

Layer	Out Channels	Out Resolution
ConvBlock	16	64x64
ConvBlock	32	32x32
ConvBlock	64	16x16
FC,BN,Relu	1024	
FC	3	

Table 3.1: Classification Network Architecture with the resolution of the input image 128x128

3.10 Loss Function

The loss function is used in the neural network to calculate the bias between the ground truth and the predicted value. It is a criterion for evaluating the performance of the network and can guide the optimization direction of network parameters. In this work, the following two loss functions are mainly employed.

Cross Entropy Loss
For multi-classification tasks, the softmax activation function is generally applied to normalize the output before the cross entropy loss function is applied, so that the sum of the predicted probabilities of each category is 1. The calculation formula of softmax is shown in Eq.3.4:

$$Softmax(a_i) = \frac{e^{a_i}}{\sum_{i=1}^{n} e^{a_i}} \tag{Eq.3.4}$$

Among them, i represents the i^{th} category, and a_i represents the score of category i predicted by the network.

The formula of cross entropy loss is shown in Eq.3.5. a_i is the ground truth of category i, while p_i is the probability value of category i, and N is the number of categories.

$$Loss_{CE} = -\frac{1}{N}\sum_{i=1}^{N} a_i log(p_i) \tag{Eq.3.5}$$

Binary Cross Entropy Loss
In the lane line existence prediction module, it is necessary to determine whether each lane line exists. Since it is a binary classification problem, therefore, the binary cross entropyEq.3.6 is applied.

$$Loss_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} a_i log\hat{a}_i - (1-a_i)log(1-\hat{a}_i) \tag{Eq.3.6}$$

N represents the total number of lane lines, while x_i indicates the probability of predicting whether the $i-th$ lane line exists, and $\hat{x}_i$ indicates the true value of whether the $i-th$ lane line exists.

Weighted Loss Function
In computer vision tasks, category imbalance is a common problem. Imbalanced categories can easily cause the model to focus on categories with a

large number of samples, while ignoring categories with a small number of samples. In view of this situation, different weights can be set for the minority sample category and the majority sample category, so that the model can pay more attention to the minor sample.

In the task of lane line segmentation, because the shape of the lane line is slender, the pixels of the lane line in the image only occupy a small area, and the rest are the background. Therefore, In order to reduce the weight of the background category, the weighted cross entropy loss function is introduced in this work, so that the model training pours more attention into the lane line part.

Chapter 4

Experiments and Results

4.1 Experimental Environment

The training, testing and evaluation of this project are all done on the servers of Boden Intelligent Technology, LLC. The experimental environment is Ubuntu operating system, and the GPU model is NVIDIA GeForce GTX 2080Ti. There are 6 GPUs in total, the video memory size of each piece is 11GB. The characteristics listed in Tabel 4.1.

All the code is implemented based on the PyTorch deep learning framework.

OS	Ubuntu 16.04
CPU Model	Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz
Computing Cores	48
Number of Nodes	2
Sustained Performance	1920 FLOPs
GPU Model	NVIDIA GeForce RTX 2080Ti Graphics Card
Number of GPUs	6
Single GPU Memory	11GB

Table 4.1: Configuration information of server of Boden Intelligent Technology, LLC.

4.2 Implementation Details

4.2.1 Pre-processing

First of all, we cut off the sky part of the images because the lane line will not appear in the sky. Then we resized the new input images to 208x976. The smaller size of the input image is, the more efficiency can we get from the inference model in real-time performance.

As to data augmentation, in this work, random scale, random crop, random rotation, random flip, random erasing and normalize are used to process the input images. During the experiments, we set the same data augmentation strategies to the method.

4.2.2 Training Hyper-parameters

The entire training process is divided into 2 steps. First, trained the semantic segmentation module, a total of 15 epochs, and 50000+ iterations. After the training of the semantic segmentation module is completed, the lane line classification branch is trained.

We apply SGD[27] optimizer to update all layers in the model and the initial learning rate is set to 0.01. The learning rate update strategy is exponential decay:

$$lr = lr_{init} * (\frac{1 - epoch}{epoch_{total}})^{0.9} \quad \text{(Eq.4.1)}$$

The total number of training epochs is 15 and the batch size is set to 20. Because of the slender shape of the lane line, the pixel area occupied by it is very small in the whole image. Therefore, when classifying each pixel, a weighted cross entropy loss function is applied. Among them, the weight of the background part is set to 0.4 to reduce the impact of category imbalance.

4.2.3 Feature Extraction

Due to the slender shape of the lane line, it only occupies a small amount of pixels in the image. Therefore, it is necessary to pre-process the road image to eliminate the noise that affects the lane line detection. After that, highlighting the contour information of the lane line and improve the accuracy of the subsequent algorithm.

In the field of computer vision, according to different detection tasks, a part of the image is generally taken as the area to be processed. This part of the area is also called the Region of Interest (RoI). In the lane line classification task, what we need to predict is the type of the lane line directly in front of the vehicle. When dividing the area of interest, it is necessary to remove the areas that interfere with the classification results, such as the sky, buildings, and the hood of the ego-car. The robustness of the algorithm can be improved after the interference is effectively eliminated. Moreover, reducing the image size can speed up the inference speed of the model.

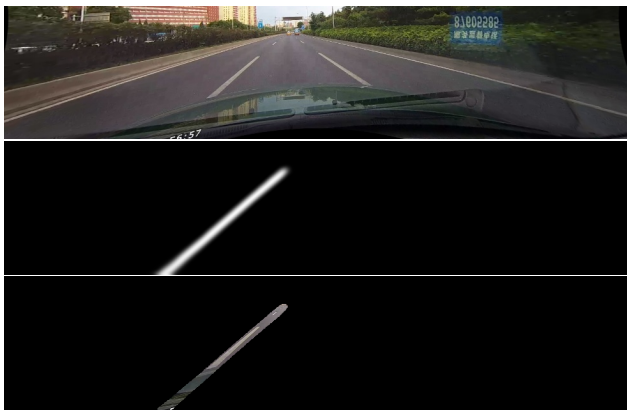


Figure 4.1: Example of feature extraction of left lane line. From top to bottom are the original image, the segmentation image, and the lane line extraction image.

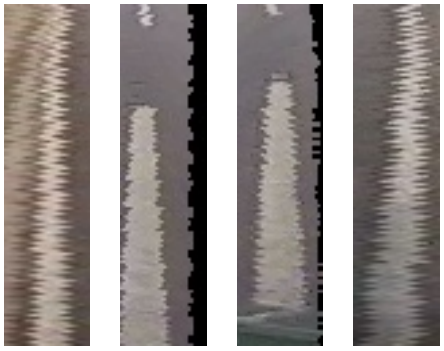
The steps for feature extraction of lane lines are as follows:

1) Since the lane line will only exist on the road, the sky in the original image is cropped first.

2) According to the segmented image of each lane line, the corresponding pixel need to be found out in the original image.

3) Move all non-zero pixels to the leftmost side of the image.

4) Since the type of lane line in the picture is determined by the type of lane line directly in front of the hood of the vehicle. Therefore, the image needs to be further cropped, and finally cropped to a picture with a height of 128 pixels and a width of 32 pixels.



(a) LL (b) L (c) R (d) RR

Figure 4.2: Outputs after feature extraction

4.3 Evaluation Metrics

The CULane dataset uses F1-measure to evaluate the ability of the model to predict lane lines.

4.3.1 Intersection-over-Union

To evaluate the accuracy of the predicted lane lines, the first step is to calculate the Intersection-over-Union (IoU). In the semantic segmentation task, IoU is the overlap rate between the segmentation result and the ground truth.

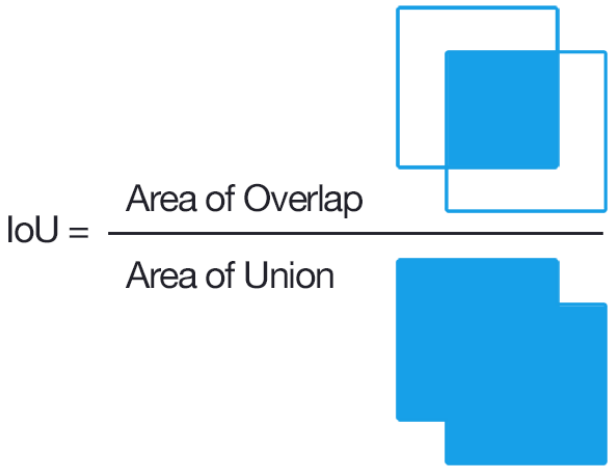


Figure 4.3: IoU equation

4.3.2 Confusion matrix

Confusion matrix is the most basic and intuitive index to measure the accuracy of classification model.

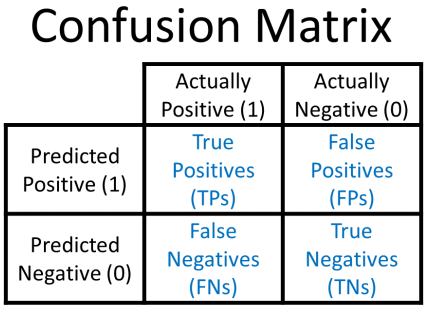


Figure 4.4: Confusion Matrix

In the lane line segmentation task:

- **TP** : Lane lines detected, the number of the correct predictions.
- **FN** : The number of lane lines that are not detected but actually exist.
- **FP** : Lane lines detected, the number of the wrong predictions.
- **TN** : This situation is not considered for lane line detection.

4.3.3 Precision, Recall and F1-Measure

With this confusion matrix, we can calculate the Precision, Recall and F1-Measure. The calculation formulas are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{Eq.4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{Eq.4.3}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{Eq.4.4}$$

The *P*recision represents the precision of the model and is the ratio of the number of correct predictions to the total number of predictions.

The *R*ecall is the ratio of the number of targets detected in the data to the total number of targets actually present.

The *P*recision and *R*ecall cannot reflect the real level of model capability, so $F1$-Measure should be used for comprehensive consideration. $F1$-Measure is the weighted harmonic average of *P*recision and *R*ecall.

4.3.4 Lane line model evaluation steps

The lane line model evaluation process is as following:

1) Get the predicted segmentation diagram of lane line by neural network;

2) Sampling at a fixed height of the segmentation diagram to obtain the sampling point of each lane line;

3) Connect the sampling points of each lane into a line with a width of 30 pixels;

4) The lines connected by the sampling points were compared with the ground truth, then calculate the IoU between them. If the IoU is larger than a certain threshold, we considered the predicted lane line was correct, which was True Positive(TP). Otherwise, it is False Positive(FP);

5) Count TP, FP and FN. Finally, calculate F1-Measure.

4.4 Results

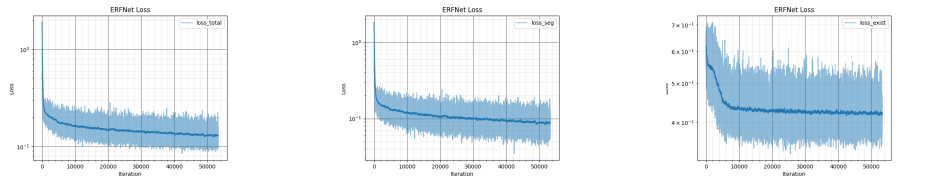
To validate our method, we will show the experimental results from two parts.

4.4.1 Performance on lane segmentation

In the lane line segmentation task, the loss formula of this project is as Eq.4.5:

$$Loss_{tot} = \alpha Loss_{seg} + \beta Loss_{exist} \tag{Eq.4.5}$$

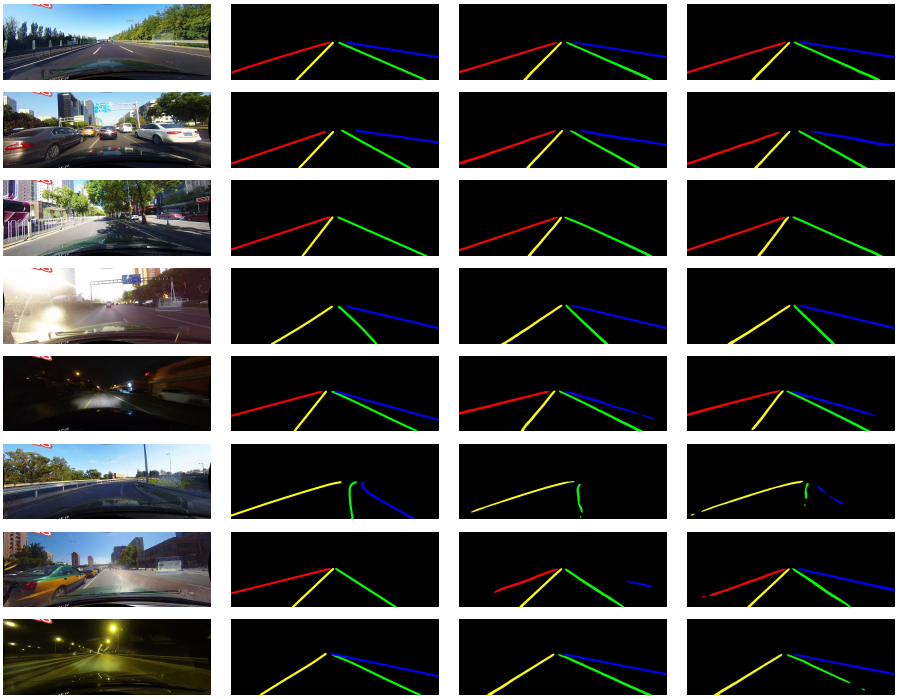
Among them, $Loss_{seg}$ uses the Cross Entropy loss while $Loss_{exist}$ uses the Binary Cross Entropy loss. Figure 4.5 shows the loss during the training process with a total of 500000k iterations. In this experiment, we set α equal to 1, while β equal to 0.1



(a) Total Loss (b) Segmentation Loss (c) Lane Existence Loss

Figure 4.5: Segmentation Loss

It can be observed from the Figure 4.5 that due to the loading of the pre-trained model, the losses drop very quickly at the beginning. The $Loss_{seg}$ reaches the optimum when it is close to 50,000 iterations. Due to the simple data structure of the lane line existence prediction branch, $Loss_{exist}$ converges quickly and is close to the optimum after 10000 iterations of training.



(a) Image (b) Ground truth (c) ERFNet (d) Our model

Figure 4.6: Visualization on the CULane dataset.

The results of the lane segmentation on the CULane datasets are given in Figure 4.6. From left to right are the origin image, the ground truth, the prediction of ERFNet model, the prediction of our model. Using different colors to indicate lane lines at different positions. Among them, red represents the left lane line of the left lane, yellow represents the left lane line of the ego-lane, green represents the right lane line of the ego-lane, and blue represents the right lane line of the right lane. Comparing the ERFNet model to our model, we can see that the segmentation results of our model in various situations have a more robust performance.

4.4.2 Ablation Study

Comparison with other models

The performance of different models on the CULane testing set is shown in Table 4.2. From the results, the performance of our lane detection-classification model outperform the baseline. The F1-Measure value reached 74.0, which is higher than SCNN algorithm and ERFNet algorithm.

Category	Proportion	ERFNet[28]	SCNN[1]	Our Model
Normal	27.7%	91.5	90.6	**91.6**
Crowded	23.4%	71.6	69.7	**72.2**
Curve	1.2%	**71.6**	64.4	65.1
Night	20.3%	67.1	66.1	**69.7**
No-line	11.7%	45.1	43.4	**46.1**
Shadow	2.7%	71.3	66.9	**75.1**
Arrow	2.6%	**87.2**	84.1	85.8
Highlight	1.4%	**66.0**	58.5	64.8
Crossroad	9.0%	2199	**1990**	2320
Total	100.0%	73.1	71.6	**74.0**

Table 4.2: F1-measure performance of different algorithms on CULane testing set.

Comparison with other attention modules

In order to further verify the effectiveness of the attention module proposed in this work, we compared it with several commonly used attention modules. The ERFNet semantic segmentation network is regarded as the baseline. The insertion position of the attention module is after the last downsampling. The hyperparameters used during training are all the same, and the data augmentation used is also the same. The experimental results are shown in Table 4.3.

Model	***Precision***	***Recall***	***F1-Measure***	***Param.(M)***	***Flops(GMac)***
Baseline			73.1	**2.61**	**11.30**
+CBAM[29]	73.8	73.6	73.6	2.64	11.69
+DANet[9]	74.1	**74.2**	**74.2**	2.93	12.58
+SE[8]	74.1	72.5	73.3	**2.61**	11.57
Ours	**74.4**	73.5	74.0	2.65	11.64

Table 4.3: Comparison results of different attention modules

From the table 4.3, we can see that adding the SE module to the model has hardly effect on the parameters. However, since SE only considers the attention in the channel dimension, its performance improvement on the model is very limitedand it is only increased by 0.2 compared with the baseline F1-Measure. DANet performs better than other attention modules in segmentation performance. But compared with other attention modules, DANet brings the largest amount of parameters. Our model increases the F1-measure by 0.9 while only increasing the amount of 0.2M parameters.

Comparison of attention modules in different positions
In this project, we also compared the performance of adding attention modules to different positions on the network. In this project, we also compared the performance of adding attention modules to different positions on the network. The position where the attention module can insert is shown in the Figure 4.7. From the table 4.4, we can see that adding the attention module proposed in this article to the Encoder's last downsampling brings the most segmentation performance gain. This shows that at the end of the encoder, the network is the deepest and contains the richest semantic information. BANet can let the model focus on critical semantic information.

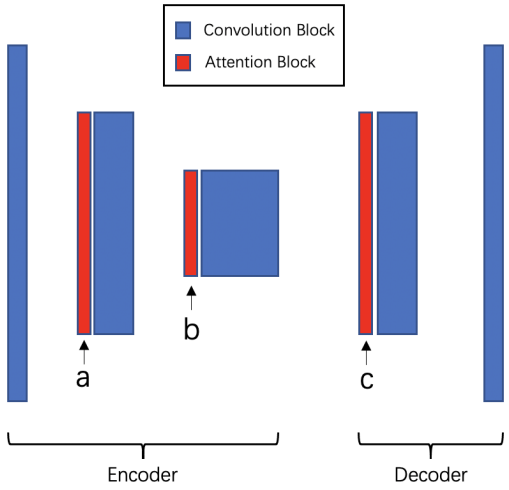


Figure 4.7: The position of the attention module in the semantic segmentation network. **a**: After the second downsampling; **c**: After the third downsampling; **c**: After the first upsampling

	a	***b***	***c***
F1-Measure	73.6	74.0	73.3

Table 4.4: Experimental results of attention modules in different positions

4.4.3 Performance on lane line classification

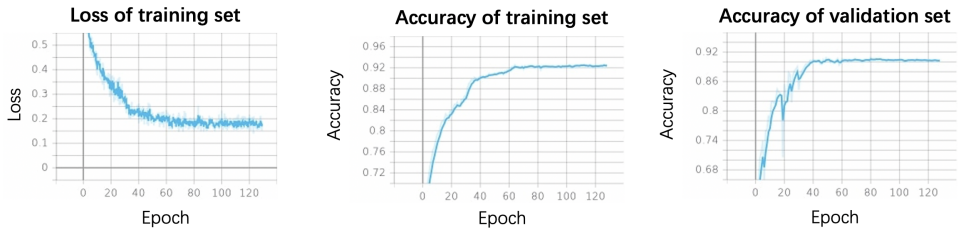
Two different experiments are carried out for the lane line classification experiments.

2 classes lane line classification

In the first experiment, we trained a binary classification network that can distinguish between dotted and solid lines. To achieve this, the lane line category information is reclassified. The *white single dotted line*, *yellow single dotted line*, *dotted-solid line*(the dotted line is close to the side of the ego-lane) are regarded as **dotted lines**. Meanwhile, *white single solid line*, *yellow single solid line*, *yellow double solid line*, *dotted-solid line*(the solid line is close to the side of the ego-lane) are regarded as **solid lines**. This part of the experiment ignores the situation that the lane features are not obvious and the lane line cannot be defined. By distinguishing whether the lane line is a dashed line or a solid line, it can be determined whether the current lane can be crossed.

Dataset	***Number***	***Lane type***	***Number***
Training	15303	*Dotted Line*	8647
		Solid Line	6656
Validation	1670	*Dotted Line*	974
		Solid Line	696

Table 4.5: Distribution of 2-class lane line dataset



(a) Loss of training set (b) Training accuracy (c) Validation accuracy

Figure 4.8: 2 classes lane line classification training curves

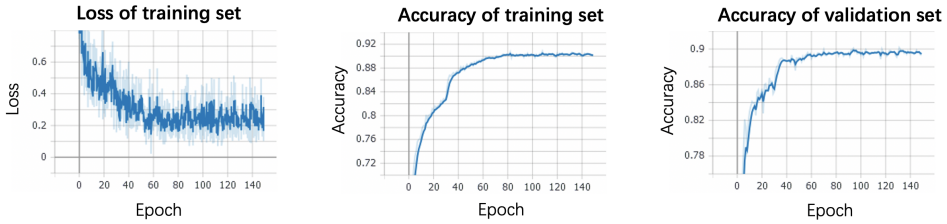
The training curves are shown in Figure 4.8. The network tends to converge in about 70 epochs. Finally, the accuracy of the two-class lane line classification model on the training set is 92.5%, and the accuracy on the validation set is 91%.

3 classes lane line classification

In the second experiment, the Undefined Line category was added. Lane lines in this case often appear at the entrances and exits of buildings. Lane lines exist but are not marked.

Dataset	***Number***	***Lane type***	***Number***
Training	16649	*Dotted Line*	8647
		Solid Line	6656
		Undefined	1346
Validation	1806	*Dotted Line*	974
		Solid Line	696
		Undefined	136

Table 4.6: Distribution of 3-class lane line dataset



(a) Loss of training set (b) Training accuracy (c) Validation accuracy

Figure 4.9: 3 classes lane line classification training curves

The training curves are shown in Figure 4.9. The training parameters are set to a total of 150 epochs, the initial learning rate is 0.001, and the loss function adopts the cross entropy loss function. The network tends to converge in about 90 epochs. Finally, the accuracy of the three-class lane line classification model on the training set is 90.7%, and the accuracy on the validation set is 90.2%. The confusion matrix for the 3 classes classification is shown in Figure 4.10.

Figure 4.11 shows the segmentation and classification results of lane lines in different scenarios. We can see that under the ideal condition, such as the light is brighter and the lane line is not sheltered, the classification prediction of the lane line is more accurate. And in the darker or more crowded scene, the accuracy of the classification result is relatively low.

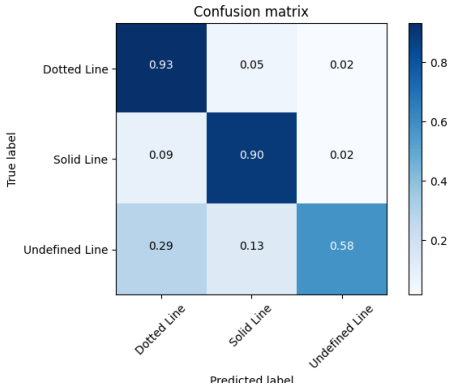
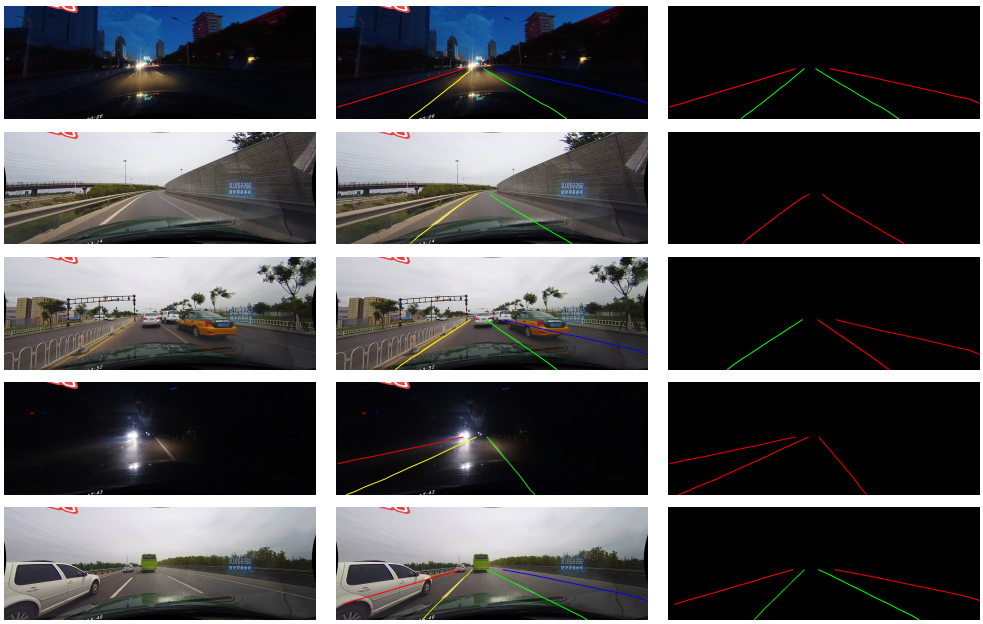


Figure 4.10: Confusion matrix for the 3 classes classification



(a) Input (b) Segmentation (c) Classification

Figure 4.11: Lane line detection and classification results of different scenes. In (c), red represents solid line and green represents dotted line.

In this case, we infer that since the input image resolution is only 128x128, it is easy to be occluded when extracting features. And there are many dark scenes in the data set, and the lane lines are not clearly displayed, which may easily cause misdetection of the classifier.

Table 4.7 is the average inference time on the test set. Among them, lane

line detection takes an average of 24ms, and lane line classification takes an average of 7ms. A total of 31ms, which meets the real-time requirements of lane line detection task.

	Detection	**Classification**	**Total**
Time(ms)	24	7	31

Table 4.7: Inference time of the model

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work, we first analyze and compare the lane line detection algorithms based on the traditional methodologies and the deep learning methodologies. Aiming at the difficulties in current lane line detection tasks, we propose a two-stage end-to-end lane line detection-classification model based on deep learning. The main contributions are as follows:

1) Labeled on CULane dataset, the largest public lane line dataset. Among them, 10600 images were selected, and the lane line classification information was added on the basis of the original annotations.

2) The Balanced Attention Network is proposed to capture the mutual dependence of global features in space and channel dimensions. While improving the segmentation performance of the model, it does not significantly increase the amount of network parameters, achieving a balance between performance and efficiency.

3) The lane line existence prediction module and the Balanced Attention Network are added to the ERFNet network. The final F1-Measure is 74.0%. Achieving comparable performance with other models with even less computation cost.

4) Designed a lane line classification network. The original image and the semantic segmentation result are combined to extract and reconstruct the image of lane line. Then use the lane line classification network to predict the category. In the two-class and three-class experiments, the accuracy on both verification sets exceeded 90%.

5.2 Future Work

The lane line detection algorithm proposed in this work has a good detection effect. Although it has been improved compared with the baseline, there are still following aspects that can be improved in future works:

1) Due to the imbalance of the data set, there are fewer scenes in the curve, which leads to the low accuracy of the model's prediction of the curve. If there are sufficient data of curve scenes for training, the robustness of the model will be further strengthened.

2) The ERFNet semantic segmentation network used in this work can only predict a fixed number of lane lines, which limits practical road applications. In the future, the network structure can be modified and corresponding post-processing can be added to predict a variable number of lane lines;

3) In the lane line classification part, we currently only divide the output into 3 categories: dashed line, solid line and non-existent. But there are more kinds of lane lines in real life scenes, and they have different meanings. In the future, if it can output more lane line classifications, it will be able to bring greater improvements to autonomous driving.

4) The collection and labeling of data sets are costly. There are few open lane line datasets and their labeling standards are different, which brings us greater difficulties in preprocessing the data. In the future, we can combine unsupervised learning methods such as GAN[30] to generate the road images we need, which can greatly reduce the cost of labeling.

Reference

[1] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial As Deep: Spatial CNN for Traffic Scene Understanding. *arXiv e-prints*, page arXiv:1712.06080, December 2017.

[2] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards End-to-End Lane Detection: an Instance Segmentation Approach. *arXiv e-prints*, page arXiv:1802.05591, February 2018.

[3] Lucas Tabelini, Rodrigo Berriel, Thiago M. Paixão, Claudine Badue, Alberto F. De Souza, and Thiago Oliveira-Santos. PolyLaneNet: Lane Estimation via Deep Polynomial Regression. *arXiv e-prints*, page arXiv:2004.10924, April 2020.

[4] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[5] Xue Bing. . https://images.app.goo.gl/7vGpKxT5WCSPC9ej6.

[6] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv e-prints*, page arXiv:1511.07122, November 2015.

[7] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv e-prints*, page arXiv:1511.00561, November 2015.

[8] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[9] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual Attention Network for Scene Segmentation. *arXiv e-prints*, page arXiv:1809.02983, September 2018.

[10] A. Borkar, M. Hayes, and M. T. Smith. Polar randomized hough transform for lane detection using loose constraints of parallel lines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1037–1040, 2011.

[11] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] Dong-Joong Kang and Mun-Ho Jung. Road lane segmentation using dynamic programming for active safety vehicles. *Pattern Recognition Letters*, 24(16):3177–3185, 2003.

[13] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer. Comparison of edge detectors: a methodology and initial study. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 143–148, 1996.

[14] Juan M. Collado, Cristina Hilario, Arturo de la Escalera, and Jose M. Armingol. Adaptative road lanes detection and classification. In Jacques Blanc-Talon, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 1151–1162, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[15] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen. A novel lane detection based on geometrical model and gabor filter. In *2010 IEEE Intelligent Vehicles Symposium*, pages 59–64, 2010.

[16] I. Fogel and D. Sagi. Gabor filters as texture discriminator. *Biological Cybernetics*, 61:103–113, 2004.

[17] A. Bosaghzadeh and S. S. Routeh. A novel pca perspective mapping for robust lane detection in urban streets. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, pages 145–150, 2017.

[18] J. Kim, S. Kim, S. Lee, T. Lee, and J. Lim. Lane recognition algorithm using lane shape and color features for vehicle black box. In *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–2, 2018.

[19] Fabio Pizzati, Marco Allodi, Alejandro Barrera, and Fernando García. Lane detection and classification using cascaded cnns. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors,

Computer Aided Systems Theory – EUROCAST 2019, pages 95–103, Cham, 2020. Springer International Publishing.

[20] Ping Luo Xiaogang Wang Xingang Pan, Jianping Shi and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *AAAI Conference on Artificial Intelligence (AAAI)*, February 2018.

[21] Tusimple datasets. https://github.com/TuSimple/tusimple-benchmark.

[22] Boden intelligent technology co., ltd. https://www.bodenai.com.

[23] Dang Ha The Hien. A guide to receptive field arithmetic for Convolutional Neural Networks. https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807, 2017.

[24] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv e-prints*, page arXiv:1412.7062, December 2014.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv e-prints*, page arXiv:1411.4038, November 2014.

[26] Eduardo Romera, Jose M. Alvarez, Luis Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–10, 10 2017.

[27] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv e-prints*, page arXiv:1609.04747, September 2016.

[28] Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, and Haowei Li. Lane Detection in Low-light Conditions Using an Efficient Data Enhancement : Light Conditions Style Transfer. *arXiv e-prints*, page arXiv:2002.01177, February 2020.

[29] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional Block Attention Module. *arXiv e-prints*, page arXiv:1807.06521, July 2018.

[30] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.