

Master's Project

by Lirui Bi

Submission date: 07-Sep-2022 10:21AM (UTC+0100)

Submission ID: 185767358

File name: 32940_Lirui_Bi_Masters_Project_430701_1672531236.docx (9.97M)

Word count: 16109

Character count: 88065



An Investigation of Lane Line Detection

by

Lirui Bi

Supervisor: **Tim Bashford**

Second Marker: **NAME HERE**

Project submitted as part of the requirements for
the award of MSc Data and Artificial Intelligence

September 2022

DECLARATION

I, Lirui Bi declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : Lirui Bi

Date : 7 September 2022

FORM OF CONSENT

I Lirui Bi hereby consent that my Project, submitted in candidature for the ***M.Sc. Data and Artificial Intelligence*** degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Lirui Bi

Date : 7 September 2022

Abstract

The solution to the problem of environmental perception is one of the important prerequisites for the realisation of intelligent vehicle autonomous driving, and lane line detection research occupies a key position in environmental perception. With the development of deep learning technology at home and abroad, and also to minimise the impact of environmental factors on the lane line detection task, researchers have started to utilise this technology for lane line detection research. This thesis considers lane line detection as an image segmentation problem and conducts research on lane line detection based on deep learning algorithms to meet the challenge of rapid environmental changes during driving.

In this work, a lane line detection method based on the U2-Net model is proposed. Firstly, in order to allow the model to focus more attention on the lane line information, a CBAM attention module is chosen to be fused on top of the U2-Net model. Then, in order to extract accurate lane position information, a position information extraction module was added to the U2-Net model. The performance of the U2-Net model in detecting lane lines was significantly improved by these two improvements. An accuracy of 95.63% was achieved on the Tusimple data, while an F1-measure of 74.4% was achieved on the CULane dataset. Experiments have shown that our method can perform lane line detection well in complex environments such as congested roads, severe obstructions, and dazzling light.

CONTENTS

1. INTRODUCTION	5
2. LITERATURE REVIEW	8
2.1 Lane Line Detection Based on Traditional Algorithms	8
2.1.1 Model-based lane line detection	9
2.1.2 Feature-based lane line detection	10
2.2 Deep learning based lane line detection	12
3. RESEARCH METHODOLOGY	17
3.1 Convolutional neural networks	17
3.1.1 Convolution layers	18
3.1.2 Activation functions	19
3.1.3 Pooling layer	22
3.1.4 Fully connected layers	24
3.2 Feature extraction network	24
3.2.1 AlexNet	24
3.2.2 VGG	25
3.2.3 GoogLeNet	26
3.2.4 ResNet	27
3.3 Semantic segmentation networks	29
3.3.1 FCN	30
3.3.2 SegNet	31
3.3.3 U-Net	32
3.3.4 DeepLabV3+	34
3.4 Dataset	36
3.4.1 Tusimple	36
3.4.2 CULane	37
3.5 Evaluation metrics	38
3.6 Experimental setup	38
4. DESIGN AND IMPLEMENTATION	40
4.1 U2Net	40
4.2 Attention mechanisms	43
4.2.1 Spatial attention mechanism	43
4.2.2 Channel attention mechanism	44
4.2.3 CBAM attention mechanism	45
4.3 Improvements to U ² -Net	46
5. RESULTS AND DISCUSSION	49
6. CONCLUSION	52
7. REFLECTION	54
REFERENCES	55

1. INTRODUCTION

Since the introduction of the automobile in the 19th century, the automobile industry has developed rapidly, and the production of automobiles has increased rapidly. The widespread use of automobiles has made life much easier. However, the number of traffic accidents has gradually increased, posing a major threat to people's lives. In order to minimise the impact of human factors on traffic, it is possible to start by improving the safety of the car, for example. Specifically, by first studying, then refining, and finally extending the safety systems of cars, the safety factor can be improved to the extent that driverless vehicles can be progressively achieved and eventually completely shielded from human influence. Based on a large amount of real-world test data, ADAS and other systems can always assist the driver when driving in order to minimise traffic accidents.

Recently, advanced assisted driving systems similar to ADAS have been emerging and are becoming a focus of research and development for use as intelligent vehicles. Such systems can provide timely alerts when the driver is driving the vehicle. The data is collected by relying on on-board sensors such as LIDAR, cameras, etc. Once processed, the system will identify the current traffic situation and assist the driver in making decisions in case of an emergency. In summary, such systems can not only make the driver more comfortable when driving the car, but also avoid traffic accidents and reduce accident injuries.

The rapid development of intelligent technology has given rise to intelligent vehicles. Since the 1870s, a few countries have put research into intelligent vehicles on the agenda. In May 1989, the NavLab series of intelligent vehicles was developed by Carnegie Mellon University in the United States, which has the significance of being the pioneering work of intelligent vehicles. The series of intelligent vehicles first rely on multiple on-board cameras to collect road data information, and then use computers to analyse the collected information and get the analysis results. In 2015, the UK also developed a smart vehicle. In the case of the Lutz Pathfinder, the vehicle can carry two people and their luggage at a speed of up to 24km/h and travel up to 64km. The intelligent vehicle relies on sensor devices such as cameras and LIDAR to obtain information about the environment around the vehicle, which is then processed by a computer for the purpose of sensing the environment and regulating the direction

and speed of the intelligent vehicle to avoid vehicles or other obstacles based on the results.

In terms of causes, there is a need to minimise or even avoid human-caused traffic accidents. In terms of conditions, artificial intelligence technology, especially deep learning technology, is currently advancing. Therefore, the need for the development of intelligent vehicle technology is just in time. In the above-mentioned intelligent vehicle driving process, the accurate completion of the task of lane line detection is crucial for driving safety. As can be seen from the above, lane line detection is of great value for the development of intelligent vehicles.

The driving environment for intelligent vehicles is complex and variable, with weather, traffic conditions, light intensity, noise interference, lane line curvature changes, and many other factors impeding driving safety. This makes lane line detection more difficult. In summary, the difficulties can be divided into the following:

(1) Bad weather. In the event of heavy snow, fog, or storms, snow or water may accumulate and obscure the lane lines, making them undetectable. In addition, the camera lens may also be disturbed by obscuring snow, fog, or rain.

(2) Complicated traffic conditions. In complex traffic conditions, there may be problems with lane lines being obscured by pedestrians or vehicles.

(3) Light intensities. Whether it is strong light, low light, or alternating between strong and weak light, lane line detection can be affected. If the light intensity changes during travel, there may be problems with reflections on the road; when travelling at night, there may be strong light from other vehicles' headlights or dim light.

(4) Noise disturbance. If long strips of noise appear on the road and are similar to other features of the lane lines, such as colour or length, false detection may perhaps occur.

(5) Curvature variation. If the curvature of the lane line changes significantly, or if the curvature is large, it will be difficult to detect.

As can be seen from the above description, the driving environment is complex and changing and is affected by many factors such as weather, the road, and traffic. In addition, lane line detection needs to take into account not only low or strong light, sudden changes in light, rain, fog, and other poor quality weather, and other environmental influences such as these, but also other unexpected conditions. In the process of engineering lane line detection, not only good accuracy but also real-time performance is required. As a result, recognition through manual feature extraction is

often difficult to cope with in complex driving environments. In addition, to address the shortcomings of the above-mentioned traditional methods that rely on manual feature extraction and deep learning-based approaches that are mostly unengineered in practice, this thesis proposes a lane line detection model based on the U²Net [1] model.

2. LITERATURE REVIEW

Many features of intelligent cars, including lane-based navigation and high-definition map modeling, are impacted by lane line detection [2]. In addition, industrial cameras have the following advantages over LIDAR detection: they are less expensive and more robust. In addition, the different stages of environmental perception can be processed by computer vision [3]. Traditional and deep learning-based algorithms are the two primary divisions of vision-based lane line identification systems. In the field of lane line detection, traditional methods usually consist of three steps: first, the image is preprocessed to eliminate noise interference, and then researchers generally choose to perform an inverse perspective transformation on the lane line image, or to delineate a region of interest (ROI). Then, feature information extraction is implemented. Finally, lane model fitting is performed to identify the lane lines. In the field of lane line detection, the accuracy of the emerging deep learning method is significantly higher compared to traditional methods. The method learns feature information autonomously and has a high generalisation capability by means of a neural network model. In recent years, in the direction of computer vision, vision-based lane line detection under the deep learning approach is generally divided into two ways: detection and segmentation. Both have in common that the preprocessed lane line images are labelled and then trained, tested, and validated in a constructed neural network model. The model is then optimised and a more accurate lane detection model is obtained. The difference is that detection refers to the classification of a determined position and only the detection target needs to be framed. The segmentation approach requires further judgement to identify which target a particular pixel belongs to.

2.1 Lane Line Detection Based on Traditional Algorithms

In order to achieve autonomous navigation of intelligent vehicles in environments such as urban and off-road, it is crucial to study lane line detection algorithms so that they are accurate, generic, and real-time [4]. As far as the traditional approach is concerned, in the model-based detection approach, the lane line detection problem is equivalent to the parameter calculation problem. The method is insensitive to noise and generally does not mis-detect lane lines if they are locally defaced and damaged in the image. Feature-based detection methods target lane line features, mark feature

points and then classify them, i.e., lane line points and non-lane line points, which are processed to obtain lane line detection results.

2.1.1 Model-based lane line detection

For model-based algorithms, there are two categories: straight line models and curve models. Due to its resistance to noise and missing data, the Hough transform is a frequently used line identification method for lane straight lines [5]. Therefore, a straight line model is often created by the Hough transform to complete the detection; for curved lanes, curve models include random sampling, consistent algorithms, and higher order curves, etc., which are fitted using several of the above commonly used methods.

For the condition of alternating straight and curved lane lines, Sui Liang et al [6] proposed a segmented model lane line recognition algorithm, which consists of a combination of the Hough transform and a quadratic curve model. First, the ROI is segmented on the image, and then the LOG operator is applied to edge detection in the ROI. Subsequently, straight lines are detected by the Hough transform and processed reciprocally to obtain two straight line segments. Based on the trend in both directions, the above two models are made to switch with the situation to finally fit the lane lines. In structured road traffic systems, curves are traffic accident prone areas. How to effectively detect lane lines and provide drivers with timely information about the traffic ahead is a difficult task to assist with safe driving. Traditional lane detection techniques are not very applicable in curvy conditions. Therefore, for the curve model, Duan et al [7] proposed a lane line detection algorithm, which was obtained by improving the sequential random sampling consistency (RANSAC). Experimental results demonstrate that this algorithm is less prone to false detection and can detect lane lines in real time and accurately when faced with a variable environment. To address the shortcomings of the existing linear model, Yanwen Wu [8] proposed an algorithm which combines vision sensors with a high-accuracy map to achieve lane line detection and tracking. In the first step, the lane line edges are obtained after improving the Hough transform; then, the selected model parameters are refreshed according to the filter prediction to complete the lane line fitting; finally, the trajectory tracking is achieved by using the high precision map and the a priori model parameters. For night driving, Li Yadi et al [9] implemented a detection algorithm for the inner lane straight line of the lane where the vehicle is located. First, the image is binarised by preprocessing, in which Laplacian operators are used to enable lane line edge

feature enhancement; then, thresholding is implemented by the OTSU algorithm and edge detection is performed by Canny operators. finally, straight line fitting is completed by the Hough transform based on slope constraints within the bottom 1/3 of the image. Using complicated road circumstances and dynamic surroundings, Cao et al. [10] suggested an intelligent vehicle lane recognition algorithm to overcome the issues of low detection accuracy and poor real-time performance of generic techniques. Firstly, the distorted image is transformed and edge detection is performed using a superposition thresholding algorithm to extract the region of interest; then, a bird's eye view of the lane line image is obtained by means of an inverse perspective transformation. The lane line curve is then fitted using the RANSAC algorithm based on the third-order B-spline curve model, and the radius of curvature is determined once the curve has been evaluated through fitting. Finally, using driving films under challenging road conditions and the TuSimple dataset, simulation test tests of the lane recognition method were conducted. Compared with other algorithms, the lane line detection algorithm has obvious advantages, with good accuracy and meets the real-time requirements. It also has strong anti-interference ability and high detection efficiency.

2.1.2 Feature-based lane line detection

Lane line features usually contain information such as colour, gradient, and intensity of edges. When feature-based lane line detection is performed, it is generally processed using pixels and filtered based on this feature to extract lane line candidate regions from the image.

Starting from colour features, Ma et al [11] proposed a detection algorithm which was implemented under the CIE Lab colour features and then clustered to achieve lane line detection. The color space can provide more valuable information than a greyscale image. The algorithm proves that it is feasible to identify lane lines by colour clustering. The quadratic curve is then used to match lanes according to the geometric characteristics of the road, and the least squares method is proposed to describe the parameters of the quadratic curve. While Li et al [12] used colour features, the image was greyed out and then the image was captured with possible noise, which was processed by median filtering. Then, the lane edges are detected by adaptive thresholding, and at the same time, the region is divided to construct a dynamic ROI, and the Hough transform is improved to obtain the candidate lines in this way. Finally, an inter-frame correlation is used to constrain the lane lines and update them

simultaneously. Some scholars focus on edge gradients, and Liu et al [13] firstly, the edge feature points are extracted by the gradient distribution characteristics of the lane line edges. Then, the lane line feature points are used to have continuity. In addition, with properties such as gradient direction consistency, clustering of feature points is implemented to obtain discrete regions. At the same time, the similarity between each region is assessed and clustering is carried out. Finally, the feature points in the optimal class are selected and lane lines are fitted. For the problem of lane line image feature intensity, Win et al [14] worked out a lane detection algorithm, which is based on the Retinex algorithm and can perform feature collection for lane line detection under different lighting conditions. In addition, the algorithm can be used to effectively reduce the computation time and accurately detect the left and right boundaries. Yu et al. [15] suggested an effective lane detection approach, which is implemented via deep feature extraction, to perform real-time lane detection in various road situations. Feature extraction from the lane lines, lane fitting, and picture preprocessing are the method's three primary steps. An aerial view of the lane line picture is first created during the preprocessing step using the inverse perspective mapping (IPM) technique, and then a Canny operator is employed to produce an edge image. In the deep lane feature extraction step, a cutting-edge lane extraction method is suggested. First, a line segment detector is used to quickly recognize line segments in IPM images (LSD). The proposed adaptive lane clustering technique is then used to group the nearby line segments produced by the LSD approach. The target lines are then separated from the other lines using the suggested local grey-scale maximum cascaded spatial correlation filtering (GMSF) algorithm. A hierarchical lane detecting method was created by Cheng et al [16] to handle both organized and unstructured road surfaces. The system first categorizes the surroundings surrounding the roads before applying the proper algorithms to various types of roads. Instead of employing a single sophisticated algorithm to cover all instances, this hierarchical structure enables relatively straightforward and effective lane recognition techniques to attain high accuracy. Pixels with lane marker colors are retrieved as feature points for environmental classification. In this study, model selection and maximum likelihood estimation of Gaussian parameters in high-dimensional feature spaces are accomplished using Eigenvalue decomposition and regularized discriminant analysis. For unstructured roads, the scene in the image is divided into regions using mean-translation segmentation. Possible candidate road boundaries are selected from the

region boundaries, and then the most likely option from the candidates is selected as the lane boundary using Bayesian rules. For structured roads, vehicles of the same color as the lane markings are first eliminated from the feature points, then inclination and turning point searches are performed to determine the boundaries, and finally, lane line detection is completed by extracting the feature points. For unstructured roads, Wang et al [17] instead used a monocular camera to capture images from both sides of the road. Among the many image processing methods, an optimised method is selected through analysis and comparison to preprocess the original image, and after detecting straight lines through the Hough transform, the lane boundaries are identified and marked.

Traditional image detection methods have a number of limitations. Firstly, the application scenario is limited. In complex road environments, road wear and tear, obstacle occlusion, changing light conditions, cloudy and rainy days can all affect the algorithm's detection effectiveness, and the robustness is poor. Secondly, the number of lane lines on the road is not fixed, but the traditional method requires fixed lanes in the first place, so it is difficult to apply to real road scenarios and has poor generalisation capability.

2.2 Deep learning based lane line detection

Traditional lane line identification algorithms need a combination of expertise and skill to manually alter the parameters in the algorithm according to various driving conditions because they are vulnerable to environmental scenarios like rain, snow, fog, and darkness. In addition, the Canny operator commonly used in traditional methods is widely used for lane line edge recognition, but the traditional Canny operator has shortcomings in adaptive capability and anti-interference capability [18]. In recent years, deep learning (including recurrent neural network) techniques have won numerous competitions in the fields of pattern recognition and machine learning [19], and their development has provided new directions for implementing lane line detection in different scenarios. The construction of a neural network is typically more intricate and consists of an input layer, an implicit layer, and an output layer. The data extrapolated from the lane line image will differ if the network layers are different. When it comes to lane images, the extracted characteristics are typically low-dimensional for the underlying network layer structure, but when the layers are added, the network layer begins to learn higher-dimensional data, such as the lane image's

color, contour, or texture features. This sub-section describes a few representative studies of lane line detection algorithms based on deep learning methods.

If the road scene is straightforward, merely the RANSAC algorithm can be used to determine lanes. However, it is challenging to finish the detection due to noise edges if the road scene is complicated, including roadside trees, fences, or crossroads, etc. Due of the noisy edges, it becomes challenging to finish the detection. In order to solve this issue, Kim et al. [20] used CNNs for lane line detection. The edge map of the region of interest is input during training of the CNN. The image with simply white lane lines against a black backdrop is utilized as the target after training. By layering the convolutional and subsampling layers of this CNN, which represent the deep structure of deep learning, noise lines are eliminated.

Vision-based deep learning lane line detection algorithms in complex scenes still have many issues to be addressed, such as marker deficiencies, overexposure, and shadows. Experimental results using DNN-related algorithms show that: The relevance of each channel in the DNN is not taken into account, and the empirical scale of the perceptual field in the network is much less than the theoretical scale. D. Xiao et al. [21] suggested an attention module (AMSC) that uses learnable coefficients to integrate self-attention and channel attention simultaneously in order to overcome these two problems. The team also suggested an attentional DNN for lane marker recognition and used AMSC to LargeFOV (modified LargeFOV). Their suggested module exhibits some characteristics, such as intrinsic parallel computing advantages that require fewer parameters and convolution operations, when compared to state-of-the-art approaches that represent pixel and channel dependencies.

Lane recognition is made more difficult by how complicated the real-world road environment is. Deep learning's explosive growth in recent years has shown how to approach and resolve this issue. Deep learning is simply concerned with the diversity of samples and is unconcerned about changes in the environment. With enough samples trained, targets can be detected and recognised. On the basis of deep learning, Chen et al [22] proposed a Mask R-CNN based lane detection algorithm, which can not only detect lanes quickly, but also achieve an accuracy of 97.9% on the TSD-Max dataset. For multi-lane, Shirke et al [23] proposed an entropy fusion-based multi-lane detection method. The main use of the proposed model is to combine the results obtained from the EW-CSA-based DCNN and the area-based iterative algorithm to provide efficient multi-lane lane detection. For different driving scenarios, Fabio

Pizzati et al [24] designed a lane line detection algorithm that is based on the ROS framework and uses mean squared uncertainty estimation to obtain better performance during the training of a multi-task CNN. The algorithm first extracts information from the street scene using two CNNs and a data fusion algorithm, then processes the information using a single CNN, and finally determines whether the information is available by comparing it with the image labels. The algorithm obtains the pixel information of the lane line image without causing inaccuracies in the detection of blank parts that could lead to changes in the lane line information.

Existing work has rarely considered the detection of lane lines in adverse weather conditions, which may degrade lane detection performance. In order to use both FCN-RGB-edge and FCN-edge techniques, Zakaria et al. [25] created a fully convolutional neural network model (FCN) with RGB and Canny edge detection as input. The three stages of model creation were data collection, platform construction, and benchmarking against current techniques and data. The proposed technique produced a 4% improvement in both cases over the original FCN-RGB image. The Canny edge detection approach collected the required data from the image effectively and ignored raindrops by treating them as noise. In conclusion, the technique may help automatic data gathering systems perform better at spotting lane lines in wet conditions. When pictures are taken in rainy weather, in addition to the presence of low light, there is the potential for distortion of the lane lines as the light is reflected by the road surface due to the wetness of the road. At night, the limited light may cause color distortion. For unfavourable driving environments such as rain and night, Seokju Lee's research team [26] proposed a unified end-to-end trainable multitasking network, VPGNet, which can handle the detection and recognition of lane lines in real time. At the same time, by building this network, this research team established a benchmark dataset for lane line marking. The 20,000 images were divided into 17 lane marking categories in four different scenarios: no rain, light rain, heavy rain, and night time, which were used to train and evaluate multiple versions of the multitasking network and to verify the significance of each task.

Li et al [27] proposed an improved codec network with an instance matching normalisation network (IB network) and an attention network structure. By replacing the pixel classification of the scene annotation with a structure that captures the content image, further optimization of this part from the idea of image style transformation is considered to address the shortcomings of batch normalisation in capturing texture in

end-to-end segmentation. To exploit the visual and appearance invariance of the encoder stage instance normalisation, an IBN layer is applied instead of the normal BN layer. The network is then given an attention mechanism to have it concentrate on lane sections. With just lanes and backdrop, this structure is suitable for two different kinds of semantic segmentation problems. For lane line and road semantic segmentation based on monocular vision, When multi-level features were combined with extended convolutional operations and FPNs in the ResNet framework, Liu et al[28] 's developed expanded feature pyramid network (FPN) with feature aggregation, or DFFA, and this deep learning model proved successful and efficient.

Some research teams have also started with instance segmentation, with researchers such as Neven [29] building a network model of the lane line detection problem as an instance segmentation problem, which can be trained end-to-end so that each of these lane lines forms its own instance. The lane line identification technique is a type of instance segmentation that can manage circumstances involving many lanes as well as simultaneous lane changes. In order to parameterize the lane instances before the lane lines are segmented, the method uses an image-based alert perspective transformation rather than a fixed "bird's eye view" transformation. By constructing a dual model based on instance segmentation utilizing the now fashionable convolutional neural network (CNN), Wang et al. [30] enhanced the conventional lane detection approach. During the capture and processing of images, the distributed computing architecture offered by edge cloud computing was utilized to increase data processing effectiveness. The real-time performance of lane detection is enhanced by the lane fitting procedure, which creates a variable matrix for successful detection in the case of slope changes.

In order to use visual information more efficiently and solve the problem of spatial information transfer on feature maps, i.e. to establish spatial relationships between pixel points, previous approaches have been based on RNN and MRF/CRF post-processing. While the performance of RNN-based approaches is not very good due to the limited range of information transfer, MRF/CRF-based approaches are too computationally expensive and their performance needs to be improved. et al. proposed a spatial convolutional neural network (SCNN) [31] to detect lane lines. The traditional layer-by-layer convolution is replaced by a slice-by-slice convolution operation within the feature map. A special convolution operation is used in the segmentation module, where information of different dimensions is aggregated by processing the sliced features and adding them one by one, which allows the

information to be passed between rows of the image. However, the slow detection speed of this method limits further development.

By enhancing the semantic segmentation effect and splitting lane instances, segmentation-based detection techniques increase the accuracy of lane line identification. The noise produced by the segmentation process is typically not taken into account by end-to-end lane identification algorithms when outputting lane instances. These techniques frequently fail to accurately detect lane line borders, which is a major issue. For instance, some of the results of smooth lane identification provide a curved appearance. In order to enable reliable and quick detection of lane lines in complicated settings, we further explore in this study how to enhance the performance of lane detection algorithms based on semantic segmentation.

3. RESEARCH METHODOLOGY

This chapter first addresses convolutional neural networks, detailing their main structures and roles, including structural layers such as the convolutional layer, activation function, pooling layer, and fully connected layer. Then four representative feature extraction network models are introduced, namely AlexNet [32], Visual Geometry Group Network (VGG) [33], GoogLeNet [34], Deep Residual Network (ResNet) [35] and other networks. Meanwhile, the theoretical knowledge of four classical semantic segmentation models, including FCN [36], U-Net [37], SegNet [38], and DeepLabV3+ [39], is presented. The content of these convolutional neural networks provides the theoretical basis for subsequent discussions related to deep learning-based lane line detection algorithms. Finally, common datasets and evaluation metrics for lane line detection algorithms are presented.

3.1 Convolutional neural networks

While traditional image processing methods rely on manually designed features, deep learning does not have such limitations and is more capable of generalisation. When raw image data is fed into a neural network, deep learning is able to learn abstract feature information incrementally, from bottom to top level. Image segmentation plays an important role in the semantic understanding of image features and refers to the process of dividing an image into a number of regions with similar properties. Recently, there has been a gradual rise in the use of deep learning to segment images, and image segmentation has evolved gradually. Convolutional neural networks are the main deep neural networks used in this thesis to implement a lane line detection algorithm based on semantic segmentation.

Convolutional neural networks (CNNs) contain at least one convolutional layer. The convolutional layer performs operations by means of convolutional operations instead of using matrix multiplication [40], and was first proposed by Alexander Waibel. The network usually has a multi-layered structure. CNNs extract features by convolutional operations, reduce parameters by weight sharing, pooling, etc. to reduce computation, and finally complete image processing tasks such as classification, detection, and segmentation. The network structure of a CNN can be seen as a hierarchical model, with the input layer, the implicit layer, and the output layer in order. The training process of CNN is briefly described as follows: first, all weights are

initialised to random values. Then, the input training image is made to pass through the above network layer structure, and the probabilities of all output classes are obtained. Additionally, for the output layer, back propagation is used to calculate the gradient of the error with respect to the weights after the total error has been determined. Finally, the gradient descent approach is used to update all weights in order to reduce output layer error.

3.1.1 Convolution layers

In the case of convolution, a neighborhood feature is obtained by doing a convolution operation on a neighborhood in the original image. Convolution is the process of selecting a convolution kernel and multiplying it with the corresponding part of the pixel values in the image to obtain the sum, which is the feature value of the image in this small region. In this context, the convolution kernel can be thought of as a matrix of weights. The convolution layer is then constructed from the eigenfaces, which contain multiple neurons. In a CNN, neurons and convolution kernels are interrelated.

After the input image, the convolutional layer is used for the feature information, which requires a convolution operation to achieve. First, the lower dimensional feature information, such as line features, edge features, corner features, etc., is captured by the lower convolutional layer. Then, using the higher convolutional layers, the above feature information is reorganised. Finally, the above steps are used to obtain high-dimensional features, which are generally more abstract. After inputting the image data, the feature values are calculated using convolutional operations while the convolutional kernel slides. Once the convolution kernels have covered the input image, the overall feature information of the input is obtained. When multiple convolutional kernels are applied in a network, they usually represent different convolutional kernels, representing different pattern features. The process of convolution is shown in Figure 1.

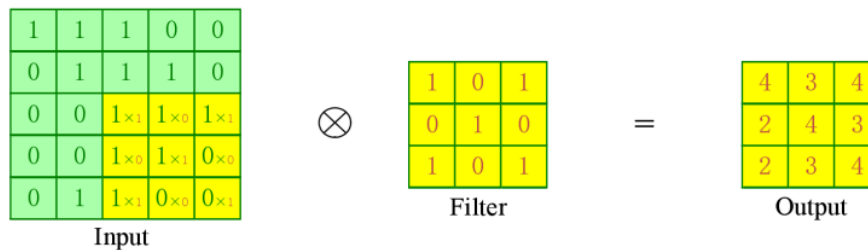


Fig. 1. Convolution operation

Assume that the input image has a width of W , a height of H , a number of channels of D , a convolution kernel of size F , a number of N , a step size of S , and a number of fill zeros of Q . After the convolution operation, the output image is defined as follows:

$$Width = (W - F + 2Q)/S + 1 \quad (1)$$

$$Height = (H - F + 2Q)/S + 1 \quad (2)$$

$$Depth = N \quad (3)$$

where *Width* indicates the width of the output image after convolution, *Height* indicates the height of the output image after convolution, and *Depth* indicates the depth of the output image after convolution.

3.1.2 Activation functions

For the activation function, which gives the neural network its core function, the ability to model non-linearities [41]. This capability is mainly achieved by discerning whether the output value of a neuron can be activated or not. For example, a neuron is considered to be activated if its output value exceeds a set threshold, otherwise it is set to 0. When the output value of a neuron is set to 0, it means that such features are weak or non-existent in that image region. When a neural network has multiple hidden layers, but no activation function, it has no non-linear representation and is no different from a neural network with only one layer. So it can be considered that adding an activation function to a neural network allows it to be layered and able to perform non-linear mapping. In the case of deep neural networks, the following types of activation functions are commonly used, and are described below.

The sigmoid activation function is defined as follows:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

The image of the sigmoid function is shown in Figure 2:

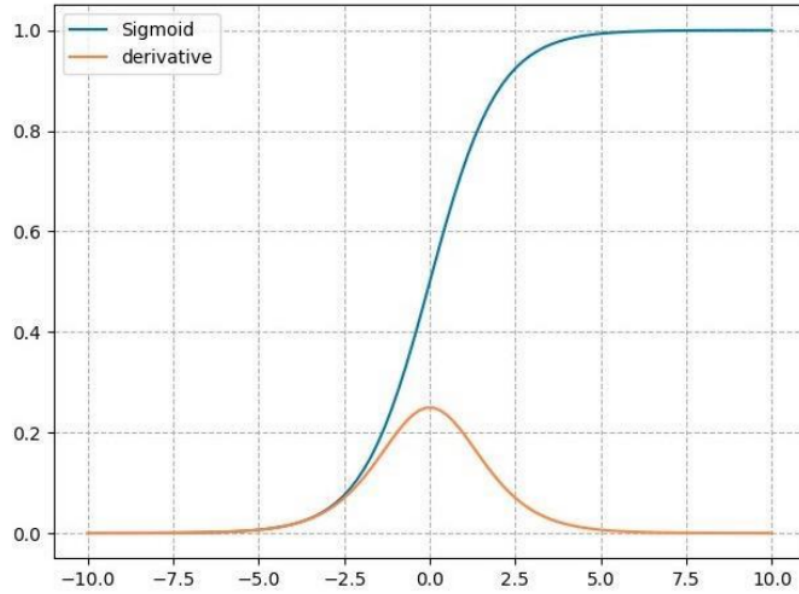


Fig. 2. Sigmoid function

The shortcomings of the sigmoid function are that, firstly, it is easy to saturate. As can be seen from Figure 2, if the input value is too large or too small, the gradient value of the neuron will gradually converge to 0. Therefore, if the initial value of the input parameter is too large and the network is propagating positively, most of the neurons will be close to saturation, which makes it difficult for the neural network to learn. This makes it difficult for the neural network to learn. If the network propagates backwards, the model weights are difficult to update. Second, the output is not 0-averaged. If the input is positive, the gradient value of the derived weight ω is also positive. When its input is negative, then the gradient sought is the same negative value. When updating the weights ω , the above situation has the following consequence: in a neural network, if it is back-propagated, the update direction is either positive or negative, which can affect the speed of convergence.

The ReLU activation function is defined as follows:

$$f(x) = \max(0, x) \quad (5)$$

The image of the ReLU function is shown in Figure 3:

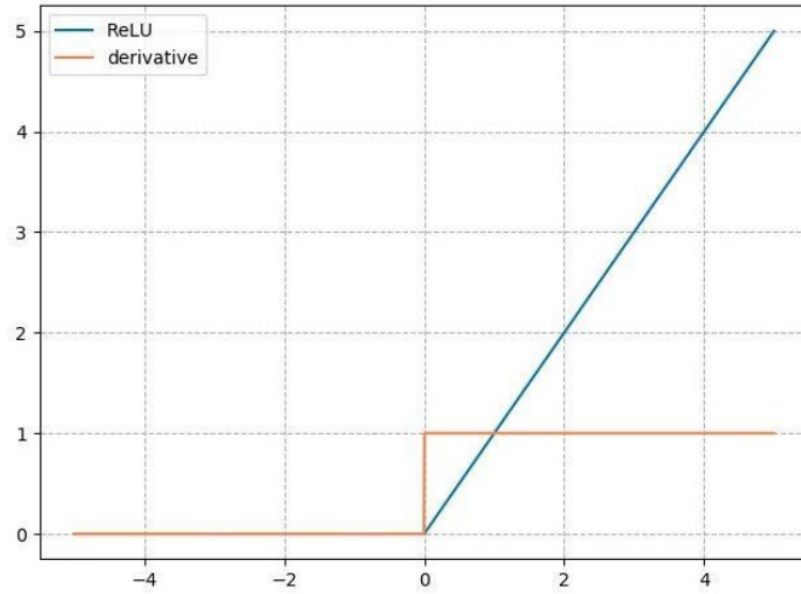


Fig. 3. ReLU function

The ReLU function is expressed in the sense that if the input of a neuron is negative, the output value is 0. Otherwise, the output is equal to the input. In backpropagation, the gradient size of the descent in this layer generally changes along with the output of the previous layer of the network, i.e., it decreases with it, so the ReLU function has the advantage of converging faster compared to the Sigmoid function. In addition, for the ReLU function, it is only required to discriminate between positive and negative output values to obtain the activation value, so it is less computationally intensive and less complex than the Sigmoid function. However, the ReLU function still has shortcomings. If the input is negative, the corresponding gradient value is 0. Therefore, when the gradient size is always 0 after the ReLU function during negative gradient calculation, and the neuron no longer responds to the data, the neuron is determined to be necrotic.

The Tanh activation function is defined as follows:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (6)$$

The image of the Tanh function is shown in Figure 3:

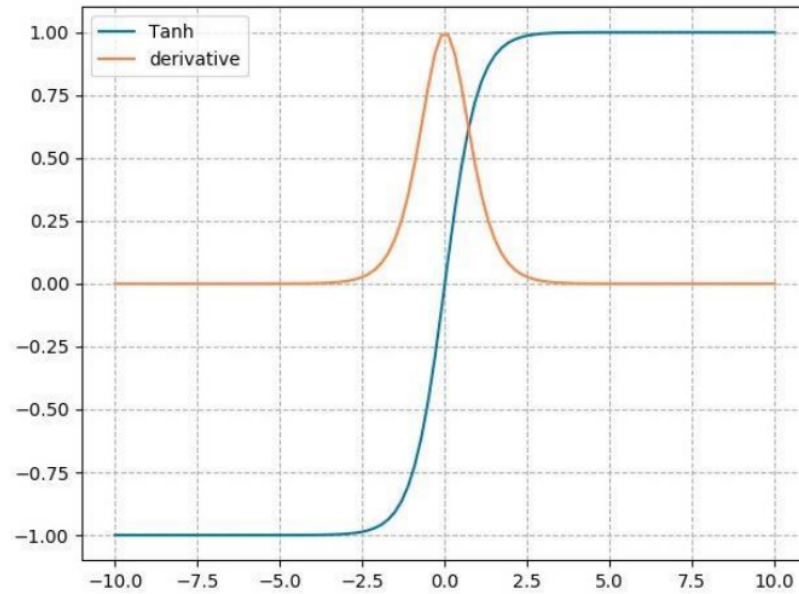


Fig. 4. Tanh function

The Tanh function maps its output values between -1 and 1 after inputting the independent variables. Therefore, the inputs of the post-layer network are almost always homogenised by 0. Compared to the sigmoid function, there is no second point of deficiency, and therefore the Tanh function is more often used. However, as with the first problem of the Sigmoid function, the Tanh function is prone to gradient saturation and this problem has never been solved.

3.1.3 Pooling layer

For Pooling Layers, which reduce the dimensionality of the data and allow the network to avoid overfitting, it is a down-sampling process [42]. This network layer integrates all the pixel features in a given neighbourhood to obtain new features. Since the convolution kernel is usually small after the convolution operation, the image is still large. In order to reduce the data dimensionality and keep the network parameters as small as possible, downsampling should be performed. After pooling, the feature information is reduced and the parameters are similarly reduced. Therefore, it is guaranteed that the features have not changed after the transformation. If the target is transformed, e.g. by translation, rotation, or stretching, the desired features are still obtained.

Pooling generally consists of mean pooling and maximum pooling. As shown in Figure 5, mean pooling involves dividing the neighborhood and averaging the selected

eigenvalues. For Max Pooling, the neighbourhood is also divided and the feature values are then averaged, as shown in Figure 6.

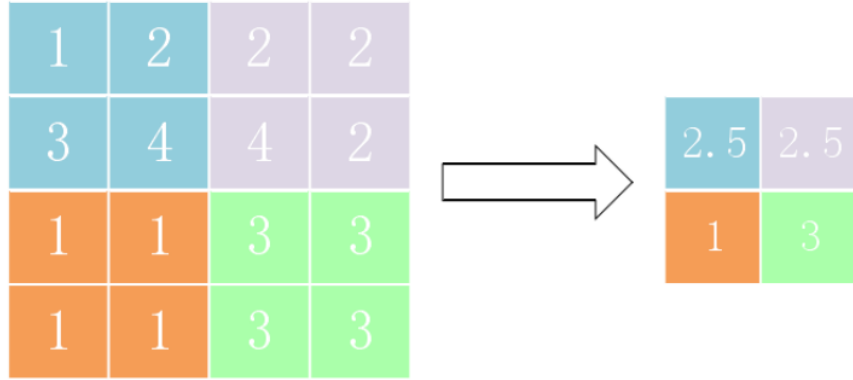


Fig. 5. Mean pooling

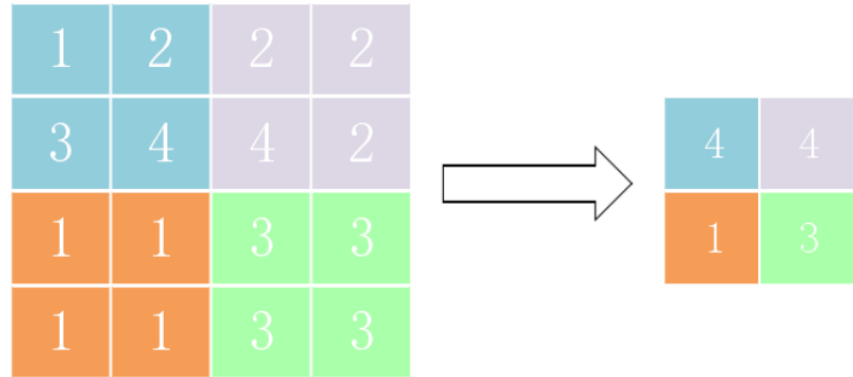


Fig. 6. Maximum pooling

Assuming that the input image has a width of W , a height of H , a number of channels of D , a pooling layer of size F and a step size S , the output image after pooling is defined as follows:

$$Width = (W - F)/S + 1 \quad (7)$$

$$Height = (H - F)/S + 1 \quad (8)$$

$$Depth = D \quad (9)$$

where *Width* indicates the width of the output image after convolution, *Height* indicates the height of the output image after convolution, and *Depth* indicates the depth of the output image after convolution.

3.1.4 Fully connected layers

Fully Connected Layers are used in CNNs for classification. In a CNN, each neuron in a fully connected layer is fully connected to the neurons in the previous layer. The information with classification properties is processed by the convolutional layer or pooling layer and then passed through the final fully connected layer. Finally, the local features are fused and their processing results are passed on to the output layer. How a suitable loss function is chosen is extremely important for a classification task. For example, when classifying in the fully connected layer, if the subject is logistic regression, it is the softmax layer. It is calculated as follows:

$$P(y = j|X) = \frac{e^{W^T X_j}}{\sum_{k=1}^K e^{W^T X_k}} - 1 \quad (10)$$

where j denotes the prediction category, X denotes the feature input from the previous layer, W denotes the weight, and K denotes the number of categories.

3.2 Feature extraction network

Feature extraction networks, also known as backbone networks, are the root of visual recognition tasks and an important part of semantic segmentation networks. Their main role in semantic segmentation networks is to extract information from images, generate feature maps, and pass them to the networks that follow. Commonly used feature extraction networks include AlexNet [32], Visual Geometry Group Network (VGG) [32], GoogLeNet [34], Deep Residual Network (ResNet) [35], etc. All these networks have strong feature extraction capabilities.

3.2.1 AlexNet

The first convolutional neural network was the LeNet-5 network, but it was the AlexNet network that first attracted attention. In 2012, Alex et al. proposed the AlexNet network, which won the ImageNet competition by a significant margin, with top-1 and top-5 error rates of 37.5% and 17.0%, respectively. The success of the AlexNet network is no coincidence. Compared with the LeNet-5 network, the AlexNet network has the following advantages:

- (1) The number of layers of the network is deepened, with five convolutional layers, three pooling layers and three fully connected layers, enhancing the feature extraction capability of the network but also posing the risk of overfitting.
- (2) The features of the ReLU activation function are exploited to avoid gradient disappearance.

(3) To suppress overfitting, the AlexNet network adopted a data augmentation strategy and also used dropout.

(4) Dual GPUs were used for training, enabling more complex convolutional neural networks and speeding up training.

The AlexNet network structure is shown in Figure 7.

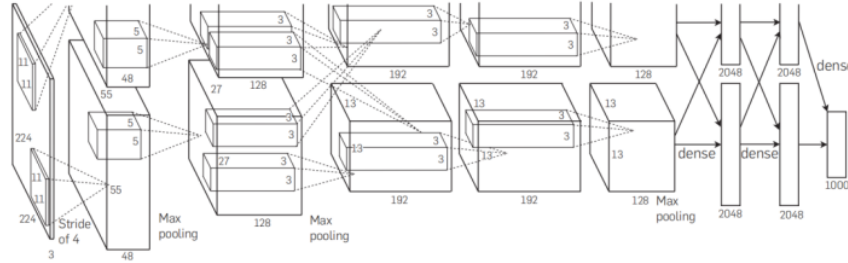


Fig. 6. The architecture of AlexNet [32].

3.2.2 VGG

The VGG [33] feature extraction network was proposed by Simonyan et al. in 2014, and the network achieved a new result in the ImageNet challenge that year, which was a significant improvement over previous methods. The VGG feature extraction network has five network structures, among which VGG16 is the most widely used, so this section focuses on the network structure of VGG16. As shown in Figure 7, the VGG16 feature extraction network consists of five modules, each containing a maximum pooling layer. Meanwhile, each module contains multiple convolutional layers, and each convolutional layer is followed by an activation layer.

The main contributions of the VGG family of networks are:

(1) Deeper convolutional neural networks were built, demonstrating that increasing the depth of the network can improve feature extraction.

(2) The convolutional layer uses smaller convolutional kernels compared to the previous versions of LeNet [49] and AlexNet [32]. The significance of using small convolutional kernels is mainly twofold. First, the convolution of stacked small convolutional kernels can obtain the same perceptual field as the convolution of a single large convolutional kernel, which significantly reduces the number of parameters of the network and improves the training speed. Secondly, the convolution of stacked small convolutional kernels can improve the non-linear representation of the network compared to the convolution of a single large convolutional kernel, making the network a better fit.

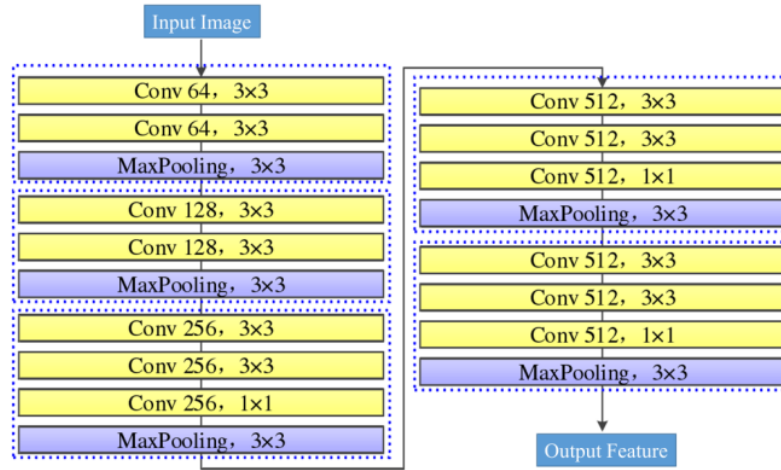


Fig. 7. The architecture of VGG.

Although the VGG family of networks has made some breakthroughs in the image classification problem, deeper networks not only increase the number of parameters and computational effort, but also introduce the problem of gradient disappearance, making the optimization of the network more difficult.

3.2.3 GoogLeNet

GoogLeNet is a deep neural network model created by the Google team based on the Inception module, which won the 2014 ImageNet competition and has since been improved over the course of the last two years. It is now accessible in various distributions, including InceptionV2, InceptionV3, and InceptionV4. Expanding the network in the deep and wide directions is the most effective way to improve the capability of neural networks, but many problems arise, such as overfitting due to too many parameters and a limited training set, high computational complexity due to large networks and large numbers of parameters, and gradient dispersion problems in deep networks. Therefore, it is considered to reduce the number of parameters but increase the depth and width of the network. Replacing full connections with sparse connections will reduce some of the number of parameters. However, most of the hardware is used to solve the density matrix, so even with very small data, the overhead of the operations cannot be effectively reduced. According to this requirement, the Google network uses the modular architecture of Inception. The essence of Inception is to combine multiple convolutional or pooling operations in one piece to form a net component, and when performing the net architecture, in the

component The construction of the net architecture is carried out in the form of a component. Figure 8 shows the detailed architecture of InceptionV1. The different scale features are fused by stitching together convolutional kernels of different sizes. However, a 5×5 convolutional kernel can still be computationally expensive when the number of layers is increased to a certain level. In order to improve this situation, GoogLeNet, like VGGNet, borrowed the 1×1 convolutional idea from Network in Network [43] and added 1×1 convolutional kernels to each of the three different sizes of convolutional kernels and maximum pooling, resulting in the complete structure of InceptionV1. The subsequent V2 version, on the other hand, used Batch Normalization [44] on top of V1 and referred to VGG with two 3×3 convolutions instead of one 5×5 convolution, as well as $1 \times n$ convolutions and $n \times 1$ convolutions instead of $n \times n$ convolutions, greatly reducing the number of parameters while causing no disturbance to the expressive power of the network. The subsequent version V3 used Label Smoothing on top of V2 and added the RMSProp optimiser. The latest version V4, on the other hand, introduced residual blocks in conjunction with ResNet.

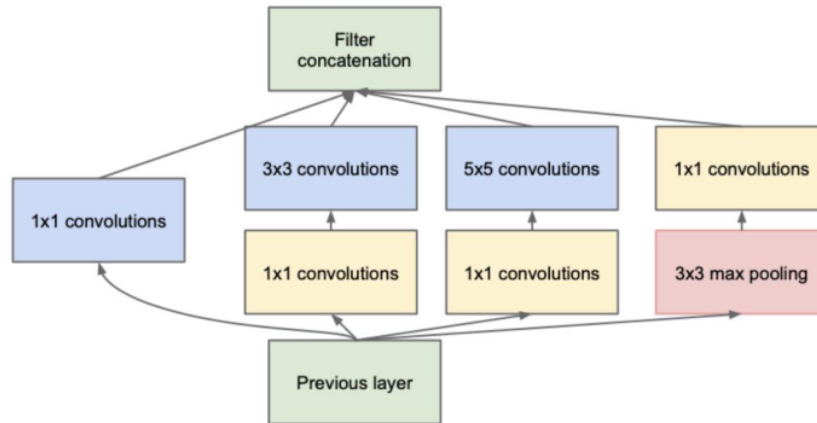


Fig. 8. The architecture of InceptionV1.

3.2.4 ResNet

As deep convolutional neural networks have been studied in depth, researchers have found that increasing the depth of the network allows the network to extract more complex features. However, deeper networks not only increased the computational effort but also caused the problems of gradient disappearance and gradient explosion during training. For this reason, in 2016, He et al. proposed ResNet [35], which won the ImageNet classification challenge in 2015, and the literature also won the CVPR Best Paper Award in 2016, which is of rapid development has important implications.

ResNet contains several network structures, and this section introduces ResNet-50 as an example, which proposes a residual block to solve the degradation problem of deep convolutional neural networks. As shown in Figure 9, the residual block consists of three convolutional layers stacked together, each containing two feature propagation paths. The residual block takes x as input, and the left branch contains three convolutional layers and two activation layers. In the left branch, the features passed into the residual block are first compressed using 1×1 convolution to reduce the dimensionality of the input features and reduce the computational effort. Feature extraction is then performed using 3×3 convolution, followed by 1×1 convolution to keep the number of output feature channels the same as the number of original input feature channels in the residual block. The right side of the residual block is the input feature constant transfer process. Finally, the features of the left and right branches are summed up to obtain the final output. The feature output (original feature mapping) of the residual block is $H(x)$, and the residual mapping obtained after the intermediate convolution layer is $F(x)$, from which we can obtain:

$$H(x) = F(x) + x \quad (11)$$

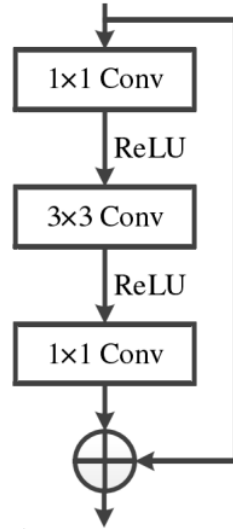


Fig. 9. The architecture of Residual Block.

The residual network causes the stacked convolutional layers to fit the residual mapping $F(x)$ instead of the original mapping $H(x)$, the reason for this operation is that it is easier to learn the residual mapping than the original mapping. When the residual mapping is 0, the residual block simply does a constant mapping at this point and the

network performance does not degrade; when the residual mapping is not 0, the residual block learns new features based on the input features, thus improving the network performance. In addition, the cross-layer connection allows features from different layers to be passed to each other, effectively alleviating the problem of gradient disappearance.

As shown in Figure 10, The ResNet network design is based on the VGG family of networks, with the addition of residual blocks through a short-circuiting mechanism and the replacement of the original VGG 3×3 convolution with residual blocks; an important design principle of the ResNet network is that the number of channels in the feature map doubles as the feature map size becomes smaller, maintaining the complexity of the network layers. At the same time, the use of convolution between ResNet modules to reduce the feature map size instead of pooling in VGG preserves the location information of the target to some extent.

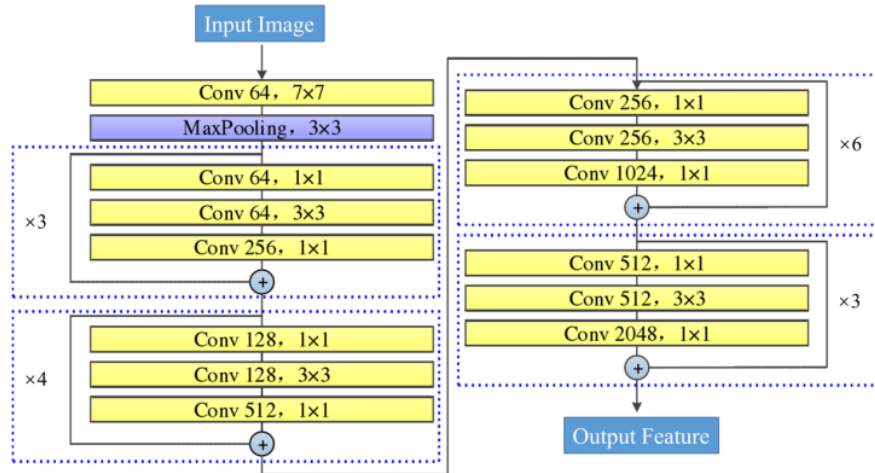


Fig. 10. The architecture of ResNet-50.

3.3 Semantic segmentation networks

Semantic segmentation of images requires that each pixel point in an image be given a specific class. When applying convolutional neural networks to the case of classification, a classification vector is usually calculated using a fully connected layer, representing the various probabilities of image classification. Therefore, the network models used for classification were not suitable for semantic segmentation of images. It was not until the advent of fully convolutional networks that this situation was broken. Since then, semantic segmentation networks have inherited the idea of full

convolutional networks, and this has led to two main types of network models for this purpose:

(1) Network models that increase the depth of the network, fuse semantic features at different levels, and aim for segmentation accuracy.

(2) Lightweight network models that use more efficient structures to build convolutional layers and are constructed with real-time performance as the goal.

3.3.1 FCN

The FCN [36], first proposed by Shelhamer Evan et al. in 2015, enables end-to-end pixel-level semantic segmentation, enabling classification of every pixel in an image, and is an inspiration for the development of subsequent semantic segmentation networks. Convolutional neural networks are suitable for image level classification and regression problems. The FCN is modified based on convolutional neural networks: the fully connected layer is replaced by a convolutional layer to achieve pixel-level semantic segmentation; the feature map of the last convolutional layer is up-sampled using a deconvolutional layer, reducing the feature map to the same size as the original image, which on the one hand classifies each image element, and on the other hand, the spatial information of the image is preserved. The FCN structure is shown in Figure 11.

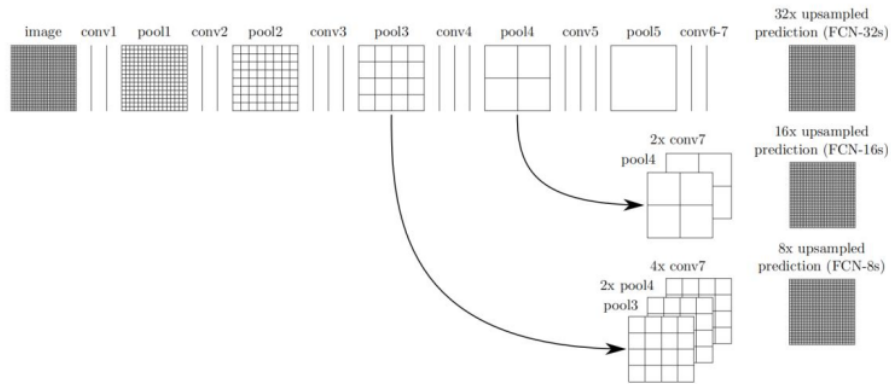


Fig. 11. The architecture of FCN [36].

The FCN replaces the fully-connected layer of the classical convolutional neural network with a convolutional layer so that the input image is no longer limited to a fixed size, thus converting the classical network into a fully convolutional neural network that can input images of any size. A second convolution and pooling operation is then performed, where the image is reduced to a quarter of the size of the original image. This is followed by a third convolution and pooling operation, where the image

is reduced to one-eighth the size of the original image, and the feature image generated by the third convolution is retained. The fourth convolution and pooling reduces the image to one-sixteenth the size of the original image and preserves the feature image generated for the fourth time. Finally, after the fifth convolution and pooling, the image has been reduced to one-thirty-second of the original image. The fully-connected layer of the convolutional neural network is replaced with a convolutional layer, and the hotspot map of one-thirty-second size after five convolutions and pooling is convolved using two convolutional layers.

The low-dimensional feature map is extracted through multiple convolutional layers to obtain a high-dimensional feature map that is rich in semantic information. In order to convert the high-dimensional feature map to the same size as the original image, FCN uses deconvolution to increase the resolution of the feature map, and the resulting hotspot map is then upsampled by a factor of 8 to restore the original image size. Since the high-dimensional feature map after feature extraction loses most of the instance information, it is difficult to recover it even after upsampling using deconvolution. The result obtained by directly increasing the resolution is known as FCN-32s, and the segmentation result is blurred with loss of detail information and low accuracy. Therefore, the feature map of conv4 is up-sampled and fused with FCN-32s to compensate for the lost detail information, resulting in FCN-16s, and the feature map of conv3 is up-sampled and fused with FCN-16s to obtain FCN-8s, which has the highest accuracy. This structure is called the "leapfrog structure", which fuses the deep semantic information with the shallow feature information and makes full use of the instance information in the low-dimensional feature map, which can obtain more detailed information and further improve the effect of semantic segmentation.

3.3.2 SegNet

SegNet, one of the classics of codec network architecture, has been an inspiration for the development of semantic segmentation models. The encoder uses downsampling to extract contextual information from the image and generate a feature map rich in semantic information. The decoder increases the size of the encoder-generated feature map, restores the location of the features, and gradually recovers the lost details, resulting in a predicted map of the same size as the original image. The SegNet network structure is shown in Figure 12.

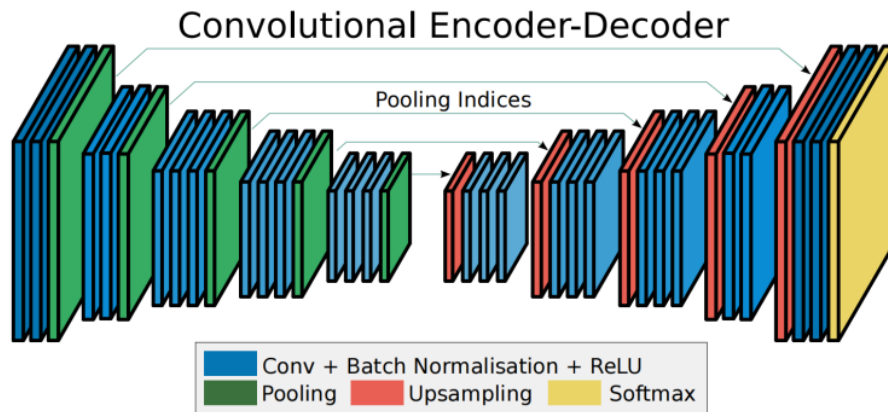


Fig. 12. The architecture of SegNet [38].

For the encoder part, SegNet uses the first 13 layers of VGG-16 to extract image features through convolutional layers. To speed up the convergence of the network, each convolutional layer is followed by a batch normalization (BN) layer and a ReLU function. To keep the image input and output dimensions consistent, SegNet fills in "0" around the input image and uses a maximum pooling operation to capture more translation invariant features.

In the decoder section, SegNet combines deconvolution and upsampling to retain detailed information on the one hand and increase the size of the image on the other. The deconvolution reduces the image classification, and the upsampling receives the pixel index values of the encoder's maximum pooling. Finally, the SoftMax classifier is used to obtain the probability of image attribute classification, which is mapped to the corresponding attribute according to the probability of the image element, completing the semantic segmentation of the image.

3.3.3 U-Net

In 2015, Ronneberger et al. proposed U-Net for medical images [37], which pushed the development of code-and-decode semantic segmentation networks. In recent years' research, it has been found that U-Net is not only applicable to medical images but also performs well in the field of remote sensing images. At the same time, U-Net solves the problem of semantic segmentation's demand for a large number of samples and computational resources and is suitable for image segmentation tasks with a small number of samples.

U-Net is a semantic segmentation model based on a fully convolutional neural network with a symmetric U-shaped structure that encodes and decodes image features

and then fuses them with deep and shallow semantic features in the network. The encoder (systolic path), on the left, consists of four convolutional blocks, each consisting of two convolutional layers with 3×3 convolutional kernels, connected to the convolutional blocks by a Max Pooling layer, with the feature map expanding to double the number of channels and reducing to one-half of its original size with each pass through the convolutional blocks and pooling layers. After the fourth convolution and pooling, the feature map is reduced to one-sixteenth of its original size, and then passes through two convolution layers of 3×3 convolution kernels and upsampling operations into the decoder. To the right of the convolution block is the decoder (dilation path), again consisting of four convolution blocks, each consisting of two convolution layers of 3×3 convolution kernels, connected to the convolution blocks by upsampling, with the feature map reduced to half its original size and increased to twice its original size with each pass through the convolution blocks and upsampling channels. After the fourth convolution and upsampling, the feature map is restored to the same size as the input image. The two sides are symmetrically distributed, forming a U-shaped structure, as shown in Figure 13. To reduce the problem of feature information loss due to maximum pooling, the output of each encoder is superimposed with the input channels of the corresponding decoder. Finally, the number of output image channels is compressed and then entered into the SoftMax classifier to obtain the probability of image element attribute classification, which is mapped to the corresponding attribute according to the corresponding probability of the image element to complete the semantic segmentation of the image.

The systolic path is responsible for capturing the image features, while the dilated path is responsible for pinpointing the features, i.e., using downsampling operations to acquire the semantic information and upsampling operations to localise the semantic information. The structure of the U-Net incorporates cropping and copying between the two sides of the path to combine deep and shallow features. The main contribution of U-Net is the use of jump-connected fusion features, which helps to restore the information loss caused by downsampling and achieve more accurate segmentation. Although U-Net is superior in simple segmentation tasks, it does not perform well in multi-category and high-resolution tasks, and it cannot achieve fine segmentation of object boundaries.

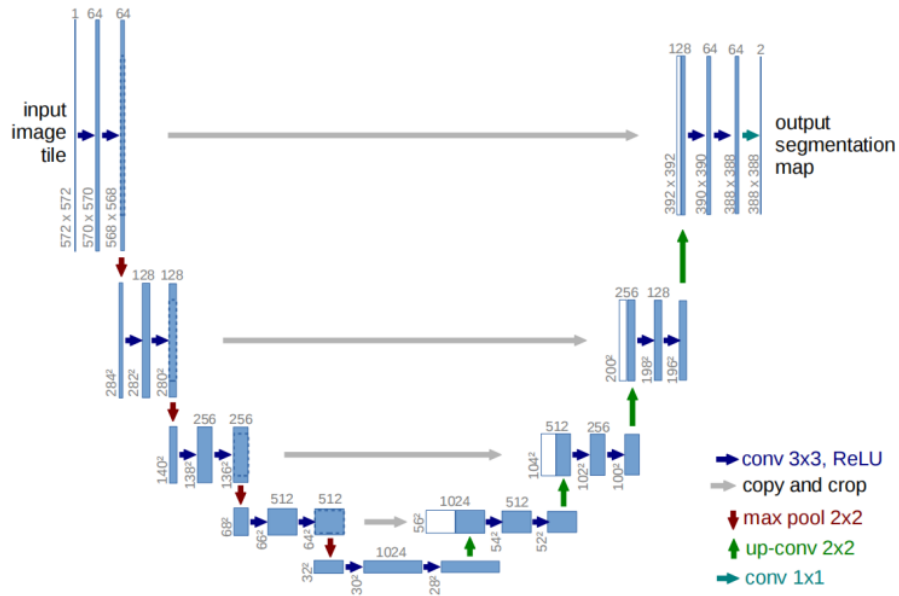


Fig. 13. The architecture of U-Net [37].

3.3.4 DeepLabV3+

The development of this direction has been provided by the hole space pyramidal pooling module and the hole convolution proposed by Chen et al. in the DeepLab series of networks [39]. The general way to enhance the perceptual field is to increase the size of the convolution kernel, but this approach increases the number of parameters and computational effort significantly, and the proposed empty convolution effectively solves this problem. The difference between this and the normal convolution is that there is a fixed distance between each convolution operator of the convolution kernel, i.e. the dilatation rate, which can be seen as a special case of the normal convolution where the dilatation rate is 1. It can be seen that the null convolution can expand the field of perception without increasing the parameters, thus obtaining more feature information.

The development of this direction has been provided by the atrous space pyramidal pooling module and the hole convolution proposed by Chen et al. in the DeepLab series of networks [39]. The general way to enhance the perceptual field is to increase the size of the convolution kernel, but this approach increases the number of parameters and computational effort significantly, and the proposed empty convolution effectively solves this problem. The difference with normal convolution is that there is a fixed distance between each convolution operator of the convolution

kernel, i.e. the expansion rate. The implementation of the null convolution is shown in Figure 14, where it can be seen that the normal convolution is a special case of the null convolution with an expansion rate of 1. It can be seen that the null convolution can expand the field of perception without increasing the parameters, thus obtaining more feature information.

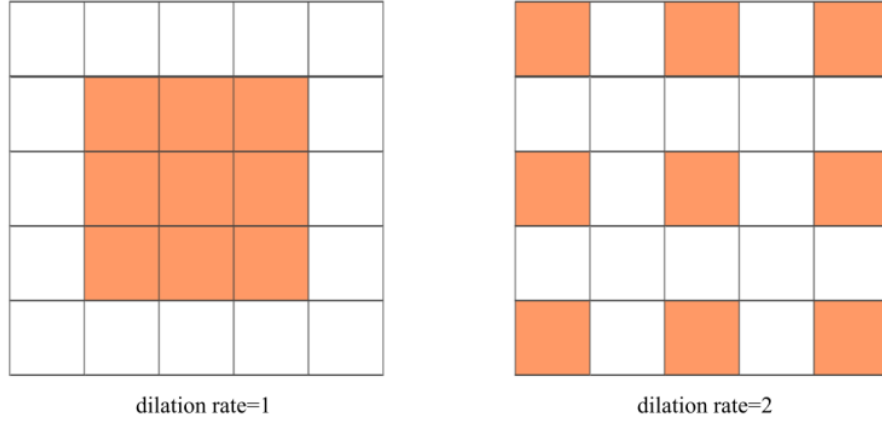


Fig. 14. Dilation Convolution.

The DeepLabV3+ [39] network is shown in Figure 15. The void space pyramidal pooling, located after the Xception [45] feature extraction network, is one of the main contributions of the Deep Lab family of networks. The module uses a parallel structure of five branches, four of which are void convolutions with different expansion rates for multi-scale information extraction, and another one is averaging pooling to integrate global information. These are simultaneously applied to the same feature map and ultimately perform channel fusion on the output of each branch. The difference with other DeepLab networks is the addition of a decoder to fuse shallow features with deep features to improve the accuracy of object boundary segmentation. Nevertheless, due to the large number of channels in the deep feature map, the parallel structure of the void space pyramidal pooling causes higher computational costs, and the problem of loss of pixel location information is not addressed.

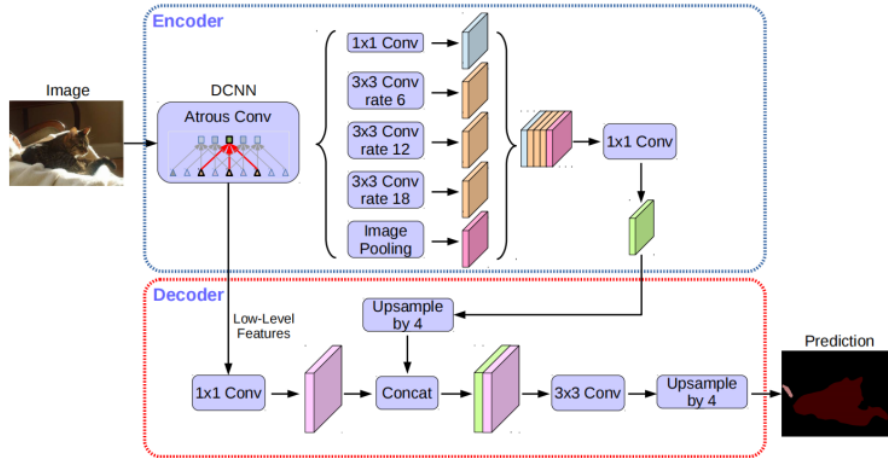


Fig. 15. The architecture of DeepLabV3+ [39].

3.4 Dataset

In deep learning, datasets are very important and the size of the dataset affects the final model's detection. The dataset is also divided into a training set, a validation set, and a test set. The model is trained on the parameters of the neural network using the training set, and the model's performance is evaluated using the validation set, which is continuously updated, tuned, and optimised. After the model has been trained, the test set is used to objectively evaluate the performance of the network. When the data in the validation set is sufficiently representative and generalized, there is no need to use a test machine.

In the field of lane line detection, there are many public datasets related to lane lines, of which the TuSimple [46] and the CULane [31] dataset are the most representative. The TuSimple dataset is mostly highway lane lines with clear lane lines and is currently more widely used in model training and new papers. The CULane dataset has a complex scene, contains a variety of complex backgrounds, and is large. The CULane dataset is complex, contains a variety of complex backgrounds, is large and challenging, and is often used for lane line detection research.

3.4.1 Tusimple

The Tusimple [46] dataset has 3626 images in the training set and 2782 images in the test set, with a resolution of 1280×720. The majority of the scenes are sunny, daytime, and lane line images at different times of the day, including 2, 3, or 4 lanes or more, with discrete, equally spaced points to represent the lane lines, and the

returned json file records the discrete The returned json file records the two-dimensional coordinates of the lane points in the diagram. The json file for each image contains discrete point information for four lane lines, i.e., the two current lane lines and the two left and right lane lines, for a total of four lane lines.



Fig. 16. Some sample data from Tusimple dataset.

3.4.2 CULane

The CULane [31] dataset is a collection of lane line data from different scenarios of a taxi in Beijing under different weather conditions. It was built for academic research on lane detection, and its acquisition time is over 50 h. The CULane dataset has a total of 133,235 frames. The number of frames in the training set is 88,880, the number of frames in the validation set is 9,675, and the number of frames in the test set is 34680 frames. The image resolution is 1640 x 590, mainly urban scenes, and the CULane dataset scenes can be divided into normal scenes and eight difficult-to-detect scenes. The challenging scenes accounted for 72.3% and included congestion, night, missing lane lines, glare, landmark arrows, dimness, curves, and intersection sections, as shown in Figure 17. The images in the dataset are all labelled with only four lane line areas, i.e., the two current lane lines and the two left and right lane lines.



Fig. 17. Some sample data from CULane dataset.

3.5 Evaluation metrics

There are various evaluation metrics for the results of lane line detection tasks, and different evaluation metrics should be used for different datasets. For the Tsuimple dataset, the official evaluation metric is accuracy. The accuracy is measured as the ratio of the number of correctly predicted lane points, C_{pred} , to the total number of lane points, T_{gt} . The higher the accuracy, the better, and the formula is as follows:

$$Accuracy = \frac{C_{pred}}{T_{gt}} \quad (12)$$

For the CULane dataset, the F1-measure, which is a weighted sum of precision and recall, is used as an evaluation metric, which combines the results of accuracy and recall. The formula is defined as follows:

$$Precision = \frac{TP}{TP + FP} \quad (13)$$

$$Recall = \frac{TP}{TP + FN} \quad (14)$$

$$F1 - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (15)$$

3.6 Experimental setup

The experimental setup in this paper mainly includes software, hardware, and various hyperparameter settings within the algorithm. The deep learning framework used is PyTorch, which is one of the mainstream deep learning frameworks with the largest user base, supports a large number of advanced APIs for deep learning, and has good compatibility. The whole experiment was run on an NVIDIA GeForce GTX 2080Ti server. The server system environment is Unbantu, and the main language used is Python 3.7. During the training process, the input data was uniformly cropped to a

size of 320×320 , the Adam optimizer was chosen to be used, the learning rate size was set to 0.0001, and the batch size was set to 8.

4. DESIGN AND IMPLEMENTATION

Lane lane detection is at the heart of assisted driving systems, and it is important to design a good detection convolutional neural network to better extract lane lane features from images. In the lane line detection task, there are problems such as missing lane lines, blurred lane lines, and occlusion by vehicles, which can seriously affect the accuracy of lane line detection. In this chapter, a U2Net-based lane detection algorithm is proposed to address these problems. First, the principles of the U²Net [1] model and the module composition are elaborated. Then, the principles of several commonly used attention mechanisms are introduced, and a suitable attention module is selected to be embedded in the U2Net model. The attention mechanism can effectively acquire rich information and extract more useful and tighter features while suppressing noisy information. Finally, as lane line detection is sensitive to boundary information, it is necessary to focus not only on the geometry of the object but also on edge information such as colour, texture, and illumination. Therefore, the U2Net model was improved by adding a module for extracting lane position information. The model was also validated by ablation experiments on the TuSimple dataset and the CULane dataset to verify the effectiveness of these improvements.

4.1 U2Net

U2net is a two-layer nested U-shaped network structure, mainly consisting of residual U-blocks (RSU) for extracting multi-scale feature extraction and an outer U-shaped structure connecting the RSU [1]. The algorithm is designed to extract higher resolution feature maps at low computational cost without using a pre-trained backbone network for image classification. The RSU consists of three parts: an input convolutional layer that performs channel conversion on the input; a U-shaped structure that extracts contextual information at multiple scales; and a residual connectivity layer that connects the input and intermediate layers via residual blocks. The RSU structure is shown in Figure 18.

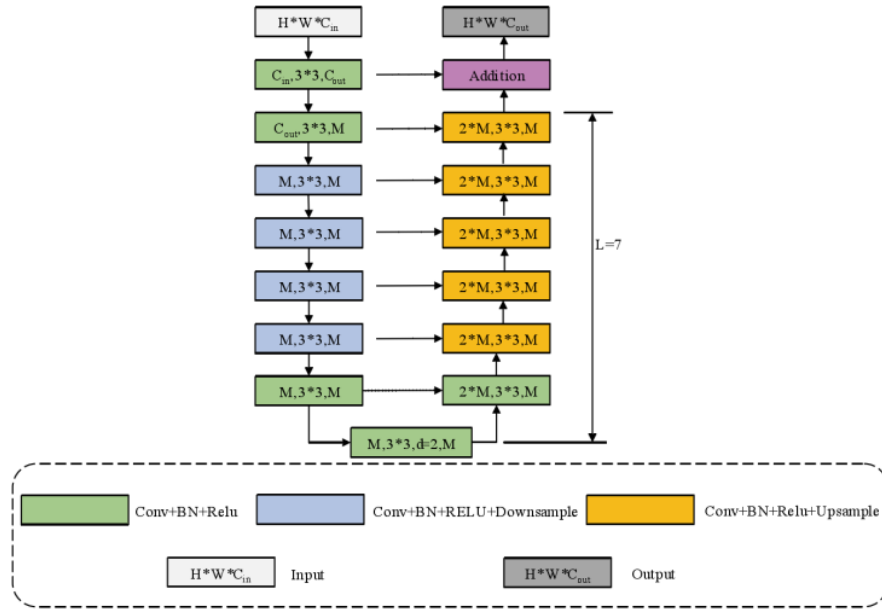


Fig. 18. The architecture of RSU.

In Figure 18, C_{in} and C_{out} are the number of channels in the input and output layers respectively, M is the number of channels in the inner layer, and H and W are the length and width of the input respectively. The larger the L , the deeper the network depth of the RSU, the more pooling operations, and the richer the local and global features extracted. The addition is the residual connectivity layer that fuses local and multi-scale features. Each green box in the diagram represents a convolutional layer, and each convolutional layer contains three operations: convolutional operations, Batch Normalization (BN), and activation function non-linearization. ReLU is a segmented linear function with one-sided inhibition, which allows sparse activation of neurons and automatic feature decoupling, improving the speed of convergence. The left side of the U-net structure is the encoding structure, where the blue box represents the convolutional layer plus downsampling operation, which obtains multi-scale features by a multi-layer 3×3 convolution operation and increases the perceptual field by a downsampling operation; the right side of the U-net structure is the decoding structure, where the yellow square represents the convolutional layer plus sampling operation, which encodes the features output from the convolutional layer into a high-resolution image. The yellow square represents the convolutional layer plus the sampling operation.

The overall structure of U²-Net consists of three parts: a 6-stage encoder; a 5-stage decoder composition; and a feature fusion output module connecting the final stage encoder and decoder, each consisting of a set of RSU modules. The overall structure of U2net is shown in Figure 19.

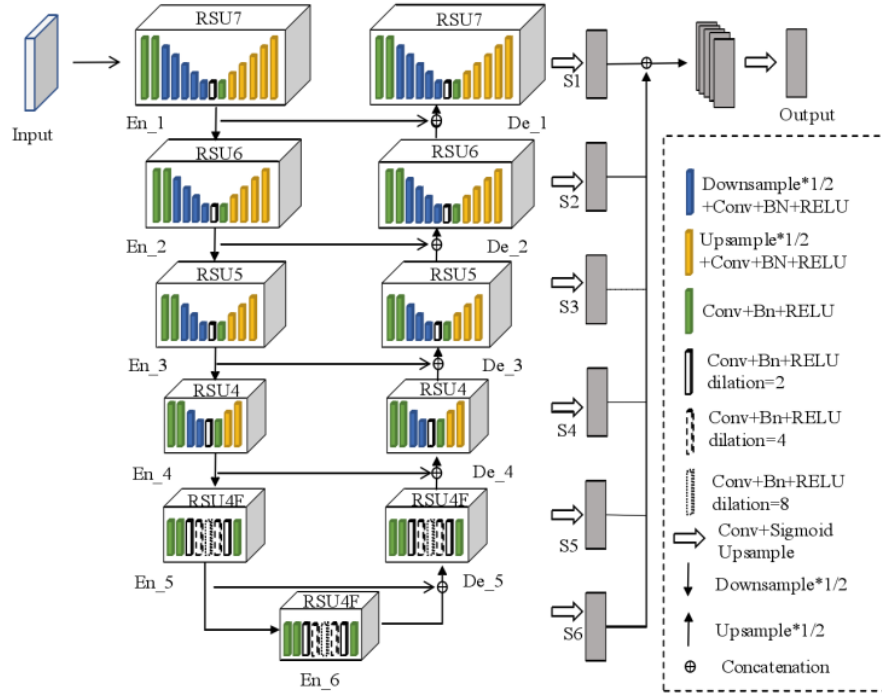


Fig. 19. The architecture of U²-Net.

In Figure 19, the first four stages of the encoder in En_1, En_2, En_3, and En_4 are filled with RSU modules with different numbers of layers, from 4 to 7, respectively, so as to obtain spatial information at different scales. In the encoders of En_5 and En_6, a modified version of the RSU-F module is used, replacing the convolutional upsampling operation with an extended convolution, so that the resolution of the feature map in the intermediate layer is the same as that of the input layer; the structure of the decoder part is similar to that of the encoder, both using RSU modules of different levels and a modified version of the RSU-F module; the final part is the significant feature map fusion module. The S6, S5, S4, S3, S2, S1 feature maps are first generated from the En_6 encoder as well as from the different stages of the decoder by means of convolution layers and sigmoid activation functions, and these probability maps are upsampled to the size of the input image. These probability maps

are then fused by a concatenation operation, a 1×1 convolutional layer, and a sigmoid function to form the final segmentation result output.

4.2 Attention mechanisms

With the rapid development of deep learning in recent years, most tasks in computer vision are based on neural networks. However, in artificial neural networks, the features cannot be equated without distinction by automatically recognising the input image. This is because different features have different strengths of influence on the inference performed by the network. The use of neural networks to construct attention mechanisms that actively focus on valid regions reduces the preference for irrelevant messages, thus improving the recognition performance of the network. In computer vision, the central task of the attention mechanism is to create a system for autonomously and independently judging the importance of messages. Using the autonomy of neurons allows the attention mechanism to have a better learning capability, thus enabling it to better understand the environment in which the neuronal network is represented and thus be able to visually optimise the network structure as well as adjust the parameters.

The basic principle of the attention mechanism can be summarised as follows: the importance of each feature is described using various weights in the network, with the important features being reinforced and the non-critical features being suppressed. In the last few years, in the field of computer vision, the attention mechanism has been mainly used to generate an attention structure model using masks, in which the neural network is trained to focus more on features in the recognition area by learning the desired area and by updating the weights and generating masks, which are then weighted with the original features.

4.2.1 Spatial attention mechanism

In the field of computer vision, the network can be made to pay more attention to the spatial location of objects in an image, and Figure 20 shows the spatial attention structure. The new feature map obtained from the channel attention is used as the input to the spatial attention module, which first performs MaxPool and AvgPool in the spatial domain to obtain two two-dimensional vectors, after which the two vectors are stitched together and convolved to obtain a two-dimensional spatial attention, which is similarly broadcast in spatial dimensions and then multiplied by the original input elements to output the final feature map. In this way, the model obtains a feature map

with richer global information, which can then be fed into the subsequent classification network to make better use of the global information to classify and predict the lane position, to a certain extent, to improve the accuracy.

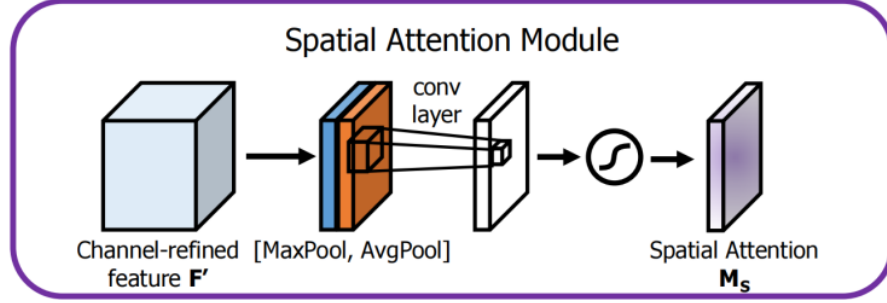


Fig. 20. The architecture of spatial attention module [47].

The specific implementation formula for the spatial attention mechanism is defined as follows:

$$F_{avg}^s = AvgPool(F) \quad (16)$$

$$F_{max}^s = MaxPool(F) \quad (17)$$

$$M_s(F) = \sigma(f^{7 \times 7}([AvgPool(F); MaxPool(F)])) \quad (18)$$

The first input feature map $F \in R^{\square \times \square \times \square}$, using maximum pooling and average pooling in parallel along the channel direction, generates two feature maps, $F_{avg}^s \in R^{1 \times \square \times \square}$ denotes the feature map obtained by averaging pooling, $F_{max}^s \in R^{1 \times \square \times \square}$ denotes the feature map obtained by maximum pooling, and the two feature maps with a channel number of 1 are fused with features in a direct stitching manner. Next, feature extraction is performed using a convolutional layer with a convolutional kernel of 7×7 , while converting the number of channels to 1 in order to bootstrap the features using multiplication. Finally, the Sigmoid function was used for activation, generating spatial attention weights of $M_s \in R^{1 \times \square \times \square}$.

4.2.2 Channel attention mechanism

In a convolutional neural network, the feature map typically has multiple channels, and after the features are extracted by the convolutional layer, each channel is not equally relevant to the global feature information. Channel attention allows the network to focus on what the image is, and Figure 21 shows the channel attention structure. The output obtained from the backbone network is first used as the input feature map in the channel domain for MaxPool and AvgPool, respectively, with the output being two one-dimensional vectors. The two are then fed into the fully

connected layer of the shared multilayer perceptron for the corresponding operations and summed to obtain the one-dimensional channel attention, which is broadcast in channel dimensions and then multiplied with the original input elements to obtain the new feature map after the channel attention adjustment.

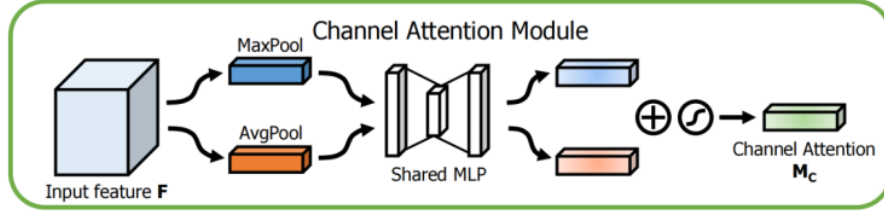


Fig. 21. The architecture of channel attention module [47].

The specific implementation formula for the channel attention mechanism is defined as follows:

$$F_{avg}^c = AvgPool(F) \quad (19)$$

$$F_{max}^c = MaxPool(F) \quad (20)$$

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \quad (21)$$

The first input feature map $F \in R^{\square \times \square \times \square}$ is generated in parallel using average pooling and maximum pooling, similar to the spatial attention mechanism to generate the feature maps F_{avg}^c and F_{max}^c , which denote the feature maps after average pooling and maximum pooling in the channel attention mechanism, respectively, where $F_{avg}^c \in R^{C \times 1 \times 1}$ and $F_{max}^c \in R^{C \times 1 \times 1}$. The two feature maps are then fed into a small shared network with the resulting feature maps at the same scale, after which the resulting feature maps are summed in a direct summation manner. The resulting feature maps are then fused in a direct summation manner. The final channel attention weights $M_c \in R^{\square \times 1 \times 1}$ are obtained by a non-linear transformation using the sigmoid function.

4.2.3 CBAM attention mechanism

The CBAM attention module fuses the spatial attention module with the channel attention module by connecting them in series, as shown in Figure 22. The fused module can focus on both spatial and channel information of the input features.

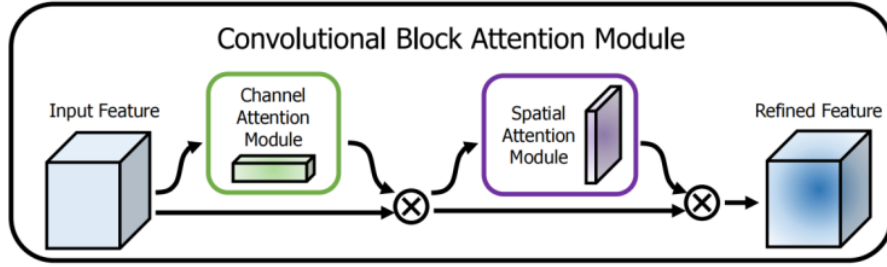


Fig. 22. The architecture of channel attention module [47].

The CBAM attention module calculation formula is defined as follows:

$$F' = M_c(F) \otimes F \quad (22)$$

$$F'' = M_s(F') \otimes F' \quad (23)$$

where F' is the channel attention output feature map, F'' is the spatial attention output feature map, i.e. the final output feature map, and $\otimes$ denotes the element-by-element corresponding multiplication operation.

4.3 Improvements to U²-Net

In deep learning methods for lane line detection tasks, convolutional neural networks concentrate on the extraction of regional features, which is a reflection of the local perceptual field of convolution. However, for complex road scenarios, the lane detection problem is difficult to solve by learning only local features. Fully-connected layers can exploit the global perceptual field in classification problems, but over-stacking fully-connected layers can easily lead to a dramatic increase in the number of parameters. In the lane detection algorithm based on line classification, the basic idea is still to use CNN for feature extraction and finally fully connected for classification. The introduction of an attention mechanism in which global information can be aggregated well and attached to local features, and the attention module itself can be lightweight and not add too much computational cost. In addition, the lane line detection task itself is based on the fact that lane lines occupy only a small number of pixels in the whole road scene, and it is clearly redundant to focus on all pixels equally. The CBAM attention module was finally chosen to be applied through ablation experiments with different attention modules.

Meanwhile, as lane line detection is sensitive to boundary information, not only the geometry of the object is concerned, but also the edge information such as colour, texture and illumination. Inspired by DFN [48], this paper also improves the U²-Net model by adding a location information extraction module (LIEM) for extracting lane

position information. The refinement residual block(RB), shown schematically in Figure 23, is where the feature mappings from the feature network's first four decoder stages all pass. A convolution layer with a 1×1 size makes up the block's initial component. The information from all channels might be combined in the interim. In order to further improve the feature map, the next is a fundamental residual block. This block's architecture, which was influenced by ResNet, can also help each stage's recognition abilities.

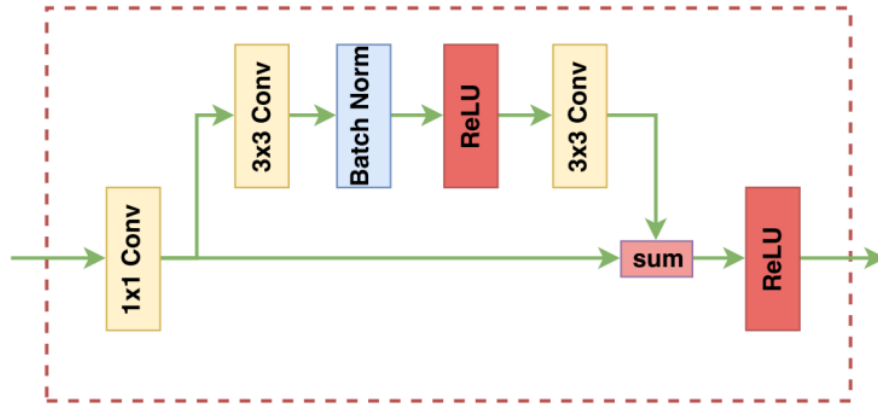


Fig. 23. The architecture of the refinement residual block(RB) [48].

When doing a semantic segmentation job, the prediction is often confused with many categories that share a similar look, especially if those categories are physically close together. As a result, we must emphasize the differentiation between characteristics. In order to motivate the feature learning, we use a semantic limit. By using explicit supervision of semantic boundaries, which forces the network to train a feature with substantial inter-class distinctiveness, we can extract precise location information. So, in order to expand the inter-class differentiation of characteristics, we suggest a spatial module. Similar to a semantic boundary detection task, it directly learns a semantic boundary with explicit semantic border supervision. This allows for the differentiation of the characteristics on either side of the semantic barrier.

There are several phases in the feature network. While the high stage features have richer semantic information, the low stage features have more detailed information. We require semantic boundaries with greater semantic implications for our task. As a result, we created the location module. Some of the original edges' lack of semantic information is eliminated since this module can concurrently get correct edge

information from low stages and semantic information from higher stages. In this manner, the high stage semantic information can stage-wise modify the low stage detailed edge information.

The overall structure of the model proposed in this paper consists of a 6-stage encoder, a 5-stage decoder, and a multi-scale feature extraction module for extracting sleep stage transition rules. The overall structure of the model is shown in Figure 24. Among them, the left side of the network is the encoding process, which consists of 6 Unet modules. The encoders of different stages are connected to each other through the Downsample layer, which reduces the network parameters while retaining the main features and prevents the model from overfitting. On the right side of the model is the decoding process, which consists of five Unet modules. In the sixth stage, the encoder En_6 and the five decoders are connected to each other via the upsample layer to recover the feature map after downsampling by a bilinear interpolation operation. At the same time, a CBAM attention module is introduced between the encoders and decoders of the first five stages, which is connected in series with the output of the CBAM module after the upsampling operation of the decoder of the previous stage is completed to obtain the decoder input of the next stage, thus achieving the effect of strengthening the learning ability of the model for important feature regions. In addition, the outputs of the first four encoders are feature refined and then fused into a feature module S7 containing rich location information. The final seven feature maps S7, S6, S5, S4, S3, S2 and S1 are fused into the final detection result map by a feature fusion operation.

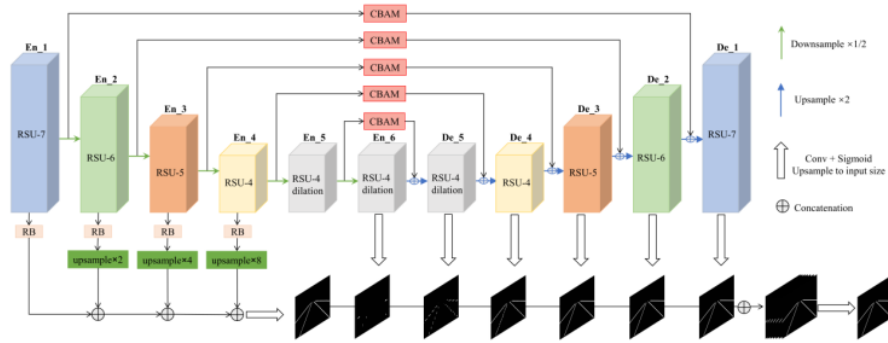


Fig. 24. The architecture of our proposed model.

5. RESULTS AND DISCUSSION

In this work, we first set up a set of ablation experiments to verify the performance of U2-Net fusing channel attention, spatial attention, and CBAM attention, respectively, as a way to select the attention combination that can better aggregate global information and improve the accuracy of lane detection. Table 1 shows a comparison of the lane detection accuracies achieved by different versions of the model on the Tusimple test set.

Method	Accuracy	Runtime(ms)
U²-Net	94.61%	19.1
U²-Net + Channel Attention	94.97%	20.5
U²-Net + Spatial Attention	95.21%	19.8
U²-Net + CBAM	95.47%	21.8

Table 1. Experiments of different attention modules on Tusimple dataset.

Based on the experimental results, it can be seen that the single channel attention module was the least effective, with a detection accuracy of only 94.97% on the Tusimple test set due to the important role of spatial location information of the lanes in detecting lane lines. In addition, fusing single spatial attention results in a higher detection accuracy of 95.21% than fusing channel attention. Finally, even better performance was achieved using the CBAM attention module, with an accuracy of 95.47% on the Tusimple test set, which was 0.86% higher than the U2-Net model without fusing any attention module. This could demonstrate a more rational network structure by having the network focus on the lane lines themselves first in the channel dimension and then on the spatial location of the lane lines in the spatial dimension through the attention mechanism. Meanwhile, the incorporation of the CBAM attention module reduces the detection rate of the model by only 2.7 ms.

Baseline	CBAM	LIEM	Accuracy
✓			94.61%
✓		✓	95.19%
✓	✓	✓	95.63%

Table 2. Experiments of different versions of the fusion model on Tusimple dataset.

By fusing the various modules after confirming the functions of the various attention modules, the function of location information extraction was confirmed. The

findings are displayed in Table 2. As shown in Table 2, the accuracy rate of the U2-Net after fusing the location information extraction was 95.19%, which is 0.58% higher than the accuracy rate of the U2-Net model before. The greatest accuracy of 95.63% was obtained when the position information extraction and CBAM attention modules were combined, surpassing U2-Net 1.02 points without any changes.

At the same time, the graphs of the results output by Grad-CAM allow a clear comparison of the attention paid to lane line information by the different versions of the model. As can be seen from Figure 25, the model that incorporates both the CBAM attention and location information extraction modules focuses most on lane line features.

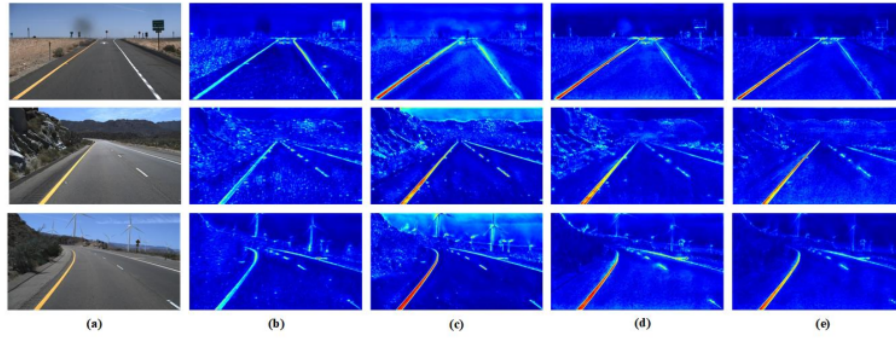


Figure 25. Heat maps of different versions of the model. (b) heat maps of U²-Net, (c) heat maps of U²-Net with spatial attention module, (d) heat maps of U²-Net with CBAM, (e) heat maps of U²-Net with both of CBAM and location information extraction module.

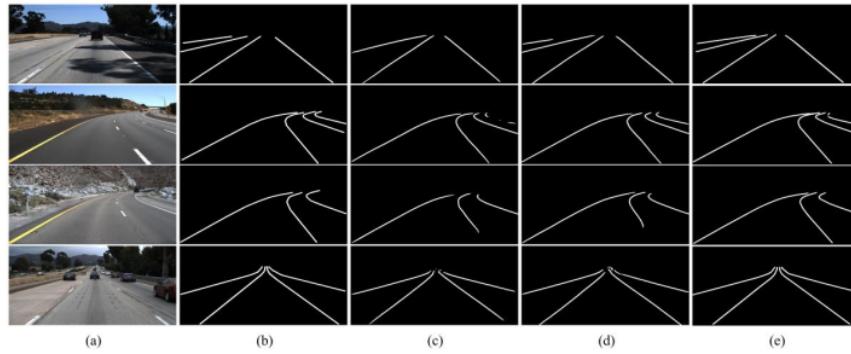


Figure 26. Lane line detection result of different versions of the model on Tusimple. (b) label (c) heat maps of U²-Net, (d) heat maps of U²-Net with CBAM, (e) heat maps of U²-Net with both of CBAM and location information extraction module.

The test results for the different versions of the model on the Tusimple dataset are shown in Figure 26. Since the Tusimple dataset is a relatively simple benchmark for lane detection, it can be seen from Figure 26 that all models of this version achieve good detection results. It is only when detecting lanes at long distances that the U2-Net, which incorporates the CBAM and location information extraction modules, can show that it can detect them more accurately.

Method	F1-measure	Runtime(ms)
U²-Net	72.1%	19.1
U²-Net + Channel Attention	72.6%	20.5
U²-Net + Spatial Attention	73.2%	19.8
U²-Net + CBAM	73.6%	21.8
U²-Net + CBAM + LIEN	74.4%	24.7

Table 3. Experiments of different versions model on CULane dataset.

Compared to the Tusimple dataset, the CULane dataset is much more complex. As can be seen from Table 3, the U2-Net, which incorporates the CABM and location information extraction modules, achieved the highest F1-measure, at 74.4%. This is 2.3% higher than the original U2-Net. From Figure 27, it is easier to compare the performance gap between the various versions of the model. Unlike other models that have difficulty detecting lane lines in complex environments, our method performs well even in complex environments such as congested roads, severe occlusions, and glare.

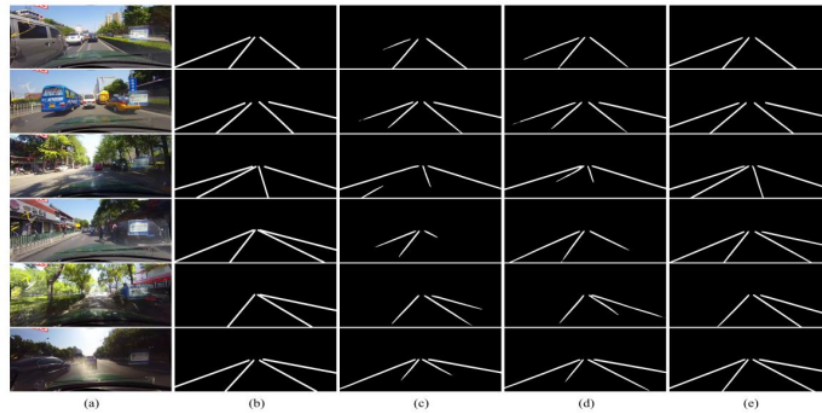


Figure 27. Lane line detection result of different versions of the model on CULane. (b) label (c) heat maps of U²-Net, (d) heat maps of U²-Net with CBAM, (e) heat maps of U²-Net with both of CBAM and location information extraction module.

6. CONCLUSION

Today, cars are commonplace in people's daily lives. The need for vehicle safety aspects is then increasing. Among the safety technologies in this area, lane line detection has an important place, and the rapid development of deep learning offers the possibility of engineering this technology in practice. Lane line detection algorithms require excellent robustness and real-time performance to enable cars to maintain fast and accurate judgements in a variety of driving environments. Current vision-based lane line detection algorithms have seen researchers move towards a deep learning-based approach, mainly through the introduction of convolutional neural networks, to deal with complex and diverse driving scenarios.

In Chapter 1, the paper first clarifies the background and significance of the research on lane line detection tasks and the problems that exist in lane line detection tasks today. Then, in Chapter 2, the research results achieved by domestic and international scholars in this field are reviewed and the future trends in this field are analyzed. Then, in Chapter 3, the basic principles of CNNs used in lane line detection models are described, mainly including convolutional layers, pooling layers, and activation functions. Relevant applications of CNNs in computer vision, including feature extraction and semantic segmentation, are also described, and several representative methods for the corresponding tasks are highlighted. In Chapter 4, the U²-Net network model is introduced and the principles, advantages, and applications of the attention mechanism are discussed. This is followed by an improvement to the U²-Net model. Relevant applications of CNNs in computer vision, including feature extraction and semantic segmentation, are also described, and several representative methods for the corresponding tasks are highlighted. In Chapter 4, the U²-Net network model is introduced and the principles, advantages, and applications of the attention mechanism are discussed. This is followed by an improvement to the U²-Net model. Firstly, the way in which the model is integrated with the attention mechanism module is described. Secondly, a location information extraction module is added to the model in order to improve the detection accuracy of the model. Finally, ablation experiments are conducted to verify the effectiveness of the improved method. The paper's lane line detection model is also tested against other methods on the dataset. It is verified that the lane line detection proposed in this paper is indeed effective in improving the

accuracy of lane detection. In particular, the F1-measure of this paper's detection algorithm achieves 74.4% on the CULane dataset. The detection accuracy on the TuSimple dataset reached 95.63%.

The main contributions of this paper are as follows:

(1) Introduction of the CBAM attention module. To address the problem that it is difficult to capture global feature information when the U2-Net-based lane line detection algorithm performs feature extraction, the attention mechanism is used to aggregate global features, which improves the accuracy of detection to a certain extent.

(2) The lane line detection is sensitive to boundary information, not only focusing on the geometry of the lane line but also on the edge information of the lane line, such as colour, texture, and illumination. In this paper, a position information extraction module is added to the U2-Net model, which focuses more on the information of the lane boundary and can better use the edge information to determine the lane lines.

7. REFLECTION

Deep learning has had a profound impact on the development of vision-based lane detection techniques, but the deep learning aspect of lane detection algorithms requires large data sets to support them, and the uninterpretable nature of neural network models leaves much to be desired in terms of stability, a problem that is also present in the deep learning aspect of the algorithms in this thesis. Even so, the deep learning-based detection algorithms are still more effective than most traditional algorithms. Therefore, this thesis argues that the future direction of lane line detection will continue to rely on deep learning to address the issues of real-time and accuracy by continuously improving the algorithms to meet the robustness requirements of intelligent vehicles at high speeds.

For the study of lane line detection, this thesis, despite the little progress made, still has many shortcomings that need further refinement and improvement:

(1) This paper does not attempt to use deep learning methods combined with traditional machine learning methods, and future attempts can be made to add other traditional machine learning algorithms again to make the algorithm more complete.

(2) The main purpose of the algorithm in this paper is to detect lane lines quickly and accurately, and no corresponding consideration has been made for the color of the lane lines or the type of virtual reality.

(3) Although this paper uses a representative data set for experiments, it is still slightly inadequate compared to real-life severe weather and complex surroundings, which makes the network's generalisation capability insufficient.

(4) Lane line detection is the core of an assisted driving system, but to ensure vehicle traffic safety, it is not enough to achieve fast and accurate lane line detection, but should also consider the integration with other modules of the assisted driving system, such as GPS, IoT, and other high-precision sensors (LIDAR, etc.), through the aggregation of information from various aspects, so as to achieve driving safety.

REFERENCES

- [1] Qin X, Zhang Z, Huang C, et al. U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection. arXiv preprint arXiv:2005.09007, 2020.
- [2] Liang D, Guo Y C, Zhang S K, et al. Lane detection: a survey with new results. *Journal of Computer Science and Technology*, 35: 493-505, 2020.
- [3] Xue J, Wang D, Du S, et al. A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars. *Frontiers of Information Technology & Electronic Engineering*, 18(1): 122-138, 2017.
- [4] Li J, Shi X, Wang J, et al. Adaptive road detection method combining lane line and obstacle boundary. *IET Image Processing*, 14(10): 2216-2226, 2020.
- [5] Fernandes L A F, Oliveira M M. Real-time line detection through an improved Hough transform voting scheme. *Pattern recognition*, 41(1): 299-314, 2008.
- [6] Sui L, Dang J, Wang Y. Fast lane line detection based on segment switching model. *Computer Applications and Software*, 34(8): 201-205, 2017.
- [7] Duan J, Li Y, Zhuang B. Research on lane line detection method based on improved SIS algorithm and sequential RANSAC. *Computer Measurement and Control*, 26(8): 280-284, 2018.
- [8] Wu Y, Zhang N, Zhou T, et al. Research on lane line detection and tracking method based on multi-sensor fusion. *Computer Application Research*, 2018.
- [9] Li Y, Huang H, Li X, et al. Nighttime lane line detection based on the Canny operator and the Hough transform. *Science Technology and Engineering*, 16(31): 234-237, 2016.
- [10] Cao J, Song C, Song S, et al. Lane detection algorithm for intelligent vehicles in complex road conditions and dynamic environments. *Sensors*, 19(14): 3166, 2019.
- [11] Ma C, Xie M. A method for lane detection based on color clustering. 2010 Third International Conference on Knowledge Discovery and Data Mining. IEEE, 2010: 200-203.
- [12] Li C, Liu H, Yuan J, et al. A real-time lane line detection algorithm based on inter-frame correlation. *Computer Science*, 2017.
- [13] Liu Y, Zhou C, Liu Y, et al. Lane line detection based on edge feature point clustering. *Science, Technology and Engineering*, 019(027):247-252, 2019.

- [14]Win S Y, Lwin H H. Lane Boundaries Detection Algorithm based on Retinex with Line Segments Angles Computation. 2018 18th International Symposium on Communications and Information Technologies (ISCIT). IEEE, 2018: 160-164.
- [15]Yu G, Wang Z, Wu X, et al. Efficient lane detection using deep lane feature extraction method. SAE International Journal of Passenger Cars-Electronic and Electrical Systems, 2017, 11(2017-01-1970).
- [16]Cheng H Y, Yu C C, Tseng C C, et al. Environment classification and hierarchical lane detection for structured and unstructured roads. IET Computer Vision, 2010, 4(1): 37-49.
- [17]Wang J, Ma H, Zhang X, et al. Detection of lane lines on both sides of road based on monocular camera. 2018 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2018: 1134-1139.
- [18]Zhang X, Chen F. Lane Line Edge Detection Based on Improved Adaptive Canny Algorithm. Journal of Physics: Conference Series. IOP Publishing, 2020, 1549(2): 022131.
- [19]Schmidhuber J. Deep learning in neural networks: An overview. Neural networks, 2015, 61: 85-117.
- [20]Kim J, Lee M. Robust lane detection based on convolutional neural network and random sample consensus. International conference on neural information processing. 2014: 454-461.
- [21]Xiao D, Yang X, Li J, et al. Attention deep neural network for lane marking detection. Knowledge-Based Systems, 2020, 194: 105584.
- [22]Chen W , Wang W , Wang K , et al. Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review. Journal of Traffic and Transportation Engineering (English Edition), 2020, 7(6):748-774.
- [23]Shirke S, Udayakumar R. Fusion model based on entropy by using optimized DCNN and iterative seed for multilane detection. Evolutionary Intelligence, 2020.
- [24]Pizzati F, García F. Enhanced free space detection in multiple lanes based on single CNN with scene identification. 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019: 2536-2541.

- [25] Zakaria N J, Shapiai M I, Fauzi H, et al. Gradient-Based Edge Effects on Lane Marking Detection using a Deep Learning-Based Approach. *Arabian Journal for Science and Engineering*, 2020, 45(12): 10989-11006.
- [26] Lee S, Kim J, Shin Yoon J, et al. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. *Proceedings of the IEEE international conference on computer vision*. 2017: 1947-1955.
- [27] Li W, Feng Q, Jialun L, et al. A lane detection network based on IBN and attention. *Multimedia Tools and Applications*, 2020, 79(23-24): 16473-16486.
- [28] Liu X, Deng Z, Yang G. Drivable road detection based on dilated FPN with feature aggregation. *IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017: 1128-1134.
- [29] Neven D, De Brabandere B, Georgoulis S, et al. Towards end-to-end lane detection: an instance segmentation approach. *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018: 286-291.
- [30] Wang W, Lin H, Wang J. CNN based lane detection with instance segmentation in edge-cloud computing. *Journal of Cloud Computing*, 2020.
- [31] Pan X, Shi J, Luo P, et al. Spatial as deep: Spatial cnn for traffic scene understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [32] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012, 25.
- [33] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [34] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015: 1-9.
- [35] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016: 770-778.
- [36] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015: 3431-3440.

- [37] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. International Conference on Medical Image Computing and Computer-Assisted Intervention. 2015: 234-241.
- [38] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(12): 2481-2495.
- [39] Chen L C, Zhu Y, Papandreou G, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation. Proceedings of the European Conference on Computer Vision. 2018: 801-818.
- [40] Bengio Y, Goodfellow I, Courville A. Deep learning. Massachusetts, USA:: MIT press, 2017.
- [41] Varshney M, Singh P. Optimizing nonlinear activation function for convolutional neural networks. Signal Image and Video Processing, 2021.
- [42] Gholamalinejad H, Khosravi H. Vehicle Classification using a Real-Time Convolutional Structure based on DWT pooling layer and SE blocks. Expert Systems with Applications, 2021, 115420.
- [43] Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv:1312.4400, 2013.
- [44] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. International conference on machine learning. PMLR, 2015: 448-456.
- [45] Chollet F. Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 1251-1258.
- [46] TuSimple. Tusimple benchmark. <https://github.com/TuSimple/tusimple-benchmark>. Accessed June, 2022.
- [47] Woo S, Park J, Lee J Y, et al. Cbam: Convolutional block attention module. Proceedings of the European conference on computer vision (ECCV). 2018: 3-19.
- [48] Yu C, Wang J, Peng C, et al. Learning a Discriminative Feature Network for Semantic Segmentation. IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1857-1866, 2018.
- [49] Le Cun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1989, 1(4): 541-551.

Master's Project

GRADEMARK REPORT

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58
