



An Improved Dual-Subnet Lane Line Detection Model with a Channel Attention Mechanism for Complex Environments

Zhong-qin Bi¹, Kai-an Deng^{1(✉)}, Wei Zhong², and Mei-ying Shan³

¹ School of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, People's Republic of China
20208007@mail.shiep.edu.cn

² No. 34 Research Institute, CECT, Guilin, Guangxi, People's Republic of China

³ Institute of Information Science and Technology, East China University of Political Science and Law, Shanghai, People's Republic of China

Abstract. Recently, many lane line detection methods have been proposed in the field of unmanned driving, and these methods have obtained good results in common conditions, such as sunny and cloudy conditions. However, these methods generally perform poorly in poor visibility conditions, such as foggy and rainy conditions. To effectively solve the problem of lane line detection in a foggy environment, this paper proposes a dual-subnet model that combines a defogging model and a lane line detection model based on stacked hourglass model blocks. To strengthen the features of important channels and weaken the features of nonimportant channels, a channel attention mechanism is introduced into the dual-subnet model. The network uses dilated convolution (DC) to reduce the network complexity and adds a residual block to the defogging subnet to improve the defogging effect and ensure detection accuracy. By loading the pretrained weights of the fog-removing subnets into the dual-subnet model, the visibility is enhanced and the detection accuracy is improved in the foggy environment. In terms of datasets, since there is currently no public dataset of lane lines in foggy environments, this paper uses a standard optical model to synthesize fog and adds a new class of foggy lane line data to TuSimple and CULane. Our model achieves good performance on the new datasets.

Keywords: Complex environment · Lane detection · Defogging · Channel attention mechanism

1 Introduction

Self-driving technology imitates human driving by making decisions and performing intelligent operations, such as gear shifting, collision avoidance, object detection, and lane departure warnings, by the car system [1]. These accurate decisions and operations made by artificial intelligence will greatly reduce the burden of human drivers and can

effectively prevent traffic accidents caused by human error. Unmanned driving technology integrates key technologies from many frontier disciplines [2, 3]. Lane line detection technology is one of the important technologies for realizing unmanned driving, and it plays an important role in autonomous navigation systems and lane departure assist systems [2, 4, 5]. The information input sensors for lane line detection generally include cameras, lidar, and global positioning systems (GPS) [6]. However, among the sensors for lane line detection, lidar has high accuracy and is not easily affected by weather; it is expensive, and GPS positioning technology is not accurate and cannot meet real-time requirements. The processing algorithm is relatively mature, and camera sensors are the main sensors for lane line detection at present. Therefore, choosing to process images and videos with cameras as sensors is more common in the current lane line detection task.

A lane line is an important traffic safety feature that has the functions of distinguishing road areas, specifying driving directions, and providing guidance information for pedestrians [7]. In the actual driving environment, the general lane line detection algorithm is sufficient because highways are in good condition and the lane markings are clear; on foggy and rainy days with bad weather conditions, the detection algorithm is often affected by light and rain. It can fail in complex urban road conditions, the lane lines can be blocked due to the shuffle of vehicles and pedestrians, the lane markings can be incomplete and faded, the shadows of trees beside the roads in the country can distort lane lines, and lane lines might not be visible on urban roads and in tunnels where the light changes rapidly. In clear conditions, for roads with obvious ups and downs, lane line detection is inaccurate [8]. The above mentioned road conditions are all problems faced in the current lane line detection task and can be divided into the following four aspects: poor road light, changes in the strong and weak light in the environment; incomplete and damaged lane markings; other objects on the road blocking the lane lines; and changes in the road slope.

The lane line detection algorithm needs to meet the requirements of detection accuracy. Although detection technology based on deep learning has achieved satisfactory results, the general lane line detection method in some harsh environments still has poor detection results. The main challenge for a generic approach, especially in foggy environments (one of the most common weather phenomena in driving scenarios), is that they often fail to detect lane lines. This is because the specific spectrum between the captured object and the camera is absorbed and scattered by very small suspended water droplets, ice crystals, dust, and other particles, reducing the effectiveness of feature extraction from these images for lane line detection [9]. To improve the performance of object detection in foggy environments, previous works often regard enhancing the visibility of foggy images as a preprocessing step. Image dehazing is beneficial not only for human visual perception of image quality but also for many systems that must operate under different weather conditions, such as intelligent vehicles, traffic monitoring systems, and outdoor object recognition systems. However, a model that combines dehazing and detection methods will have increased complexity and increased parameter numbers due to the additional dehazing task, which will eventually lead to a decrease in detection speed. In addition, training a convolutional neural network (CNN)-based detection network requires a large quantity of data. Since there is no public lane line dataset containing

images in a foggy environment, many lane line detection models cannot fully learn the lane line features in foggy environments, resulting in inaccurate results.

In this paper, we propose a new dual-subnet lane detection model to solve the problem of lane detection in foggy environments. In summary, the main contributions of this paper are as follows.

- To reduce the complexity of the dual-subnet model and reduce the number of model parameters as much as possible, we choose the lightweight AOD-Net [10] as the defogging subnet.
- To improve the dehazing effect, the subnetwork combines the channel attention mechanism to focus on extracting the features of important channels and fuses the hazy image of the input model as a residual block into three connection layers.
- In the detection subnetwork, we use a stacked hourglass network model. We introduce a channel attention mechanism into the downsampling layer of the original model to extract more features about lane lines, and dilated convolution (DC) is used to reduce the network complexity to ensure detection accuracy.
- We use the standard optical model to synthesize fog to add a new class of foggy lane lines to TuSimple and CULane. We conduct a comprehensive experiment on the new datasets to comparatively evaluate and demonstrate the effectiveness of the proposed model.

The remainder of this paper is organized as follows. Section 2 describes the related works, Sect. 3 introduces the relevant preliminary knowledge used in this paper, Sect. 3 describes the proposed method, Sect. 4 presents the experimental results and Sect. 5 describes the conclusion of this paper.

2 Related Work

In the past two decades, research on lane line detection technology has achieved good results. At present, the main methods are divided into traditional methods and methods based on deep neural networks, and another category is the combination of traditional image processing and deep learning.

2.1 Traditional Methods

Traditional methods detect lane lines by manually designing detection operators according to the characteristic morphology of lane lines and rely on feature-based [11] and model-based [12] detection methods.

Feature-based detection methods use the colour and greyscale features of a road image [13] and combine the Hough transform [14] to realize lane line detection, in which the detected element is generally a straight lane line. In addition, algorithms, such as the particle filter [15], Kalman filter [16, 17], Sobel filter [18], Canny filter [19], and finite impulse response (FIR) filter [20], are commonly used in lane line detection methods. This method can adapt to the change in road shape and has a fast processing speed, but when the road environment is complex, postprocessing is needed, which reduces

the real-time performance; when lane lines are incomplete or occluded, the detection performance of the algorithm decreases [21].

The model-based detection algorithm usually constructs a lane line curve model and regards a lane line as a straight-line model, a higher-order curve model, and so on. The principle of this method is to fit the geometric model structure of the lane line by the least square method, random sampling agreement (RANSAC) algorithm, Hough transform [8], or another method according to the geometric structure characteristics of the lane line and obtain the model parameters to create a lane line detection method. The advantage of this detection algorithm is that it can reduce the impact of missing lane lines and has better environmental adaptability; the disadvantage is that if the detected road environment is inconsistent with the present model, the detection effect will be reduced.

2.2 Methods Based on Deep Neural Networks

For the lane line detection task, the process of using deep learning technology for detection is as follows: first, establish a marked lane line dataset, then train the lane line detection network on the dataset, and finally, use the trained network for the actual lane line detection task. Since the CNN AlexNet [22] won the 2012 ImageNet Large-scale Visual Recognition Challenge (ILSVRC), CNNs have been widely used in image classification, object classification, etc., due to their sparse connections and translation invariance. Excellent results have been achieved in the fields of tracking, target detection, semantic segmentation, etc., and these results have brought new ideas to research on lane line detection. Early CNN-based methods (e.g., [23, 24]) extract lane line features through convolution operations. Lane detection methods can be divided into semantic segmentation methods, row classification methods and other methods.

Segmentation-Based Methods. This method extracts the feature data of the image, carries out the image binarization semantic segmentation, divides each pixel into the lane or the background, and filters and connects the pixels of the lane line. Finally, it is decoded into a group of lane lines on the segmentation feature map by postprocessing. For example, GCN [25] and SCNN [26] do not need to manually combine different traditional image processing techniques according to specific scenes and can directly extract accurate images from input images in more complex scenes. In SCNN, the author proposes an effective scheme specifically designed for a slender structure; however, the method is slower (7.5 fps) due to larger backbones. In addition, GAN-based methods (such as EL-GAN [27]), attention maps [28], and knowledge distillation [29] (such as SAD [30]) provide new ideas for lane detection. A self-attention distillation (SAD) module is proposed to aggregate contextual information. This module uses a lighter backbone but has high efficiency and real-time performance.

Row-Wise Classification Methods. The row-by-row classification method is a simple method for lane detection based on input image meshes. For each row, the model predicts the cell most likely to contain partial lane markings. It also requires a postprocessing step to build the lane set. This method was first introduced in E2E-LMD [31] and achieved good results. In [32], by using this method, the model loses some accuracy, but the

number of computations is small, and the detection speed is fast (up to 300 fps or more). Moreover, the global receptive field can also deal with complex scenes well.

Other Methods. PolyLaneNet [33] proposed a model based on deep polynomial regression. In this approach, the model outputs a polynomial for each lane. Although it is fast, this method is highly biased towards straight lanes compared with curve lanes due to the imbalance of the lane detection dataset. In [34], lane lines are modelled by the curve description function, and lane lines are described by predicting the third-order Bezier curve based on the Bessel curve description (curve description can be realized by a small number of control points).

3 Proposed Method

In this section, a new dual-subnet model, which combines the defogging model with the lane line detection model, is introduced. The model achieves this goal through two subtasks: visibility enhancement after defogging and lane line detection. The basic structure of the dual-subnet model is shown in Fig. 1. The entire model structure is divided into two main modules, and the lane detection subnet module is divided into a resizing module and a prediction module.

- Defog subnet module: The defog module is composed of a CNN. The defog subnet module estimates the parameters required for defogging based on the input foggy image and then uses this parameter value as the input adaptive parameter to estimate the clear image after defogging. In other words, the clean image tensor after defogging is obtained to achieve the effect of feature enhancement (visibility of lane line feature).
- Resizing module: The resizing module consists of three convolutional layers, which resize the output of $256 \times 128 \times 32$ obtained from the defog subnet to $64 \times 32 \times 128$.
- Prediction module: The prediction module consists of three stacked hourglass modules. The output branch predicts confidence, offset, and embedding features. Three output branches are applied at the ends of each hourglass block. The loss function can be calculated from the outputs of each hourglass block.

3.1 Defogging Subnet Module

The original AOD-Net model consists of a K estimation module and a clean image generation module. According to formulas (1)–(4) in [10], the K estimation module mainly estimates $K(x)$ from the input $I(x)$, and the clean image generation module uses $K(x)$ as its input adaptation parameter to estimate $J(x)$, that is, to obtain the final clean image. The K estimation module utilizes 5 convolutional layers, and the 1st to 5th convolutional layers use convolution kernels of size $1*1$, $3*3$, $5*5$, $7*7$ and $3*3$, respectively, through 3 layers. The connection layer fuses filters of different sizes to form multiscale features. As shown in Fig. 2, to make the $K(x)$ module extract more

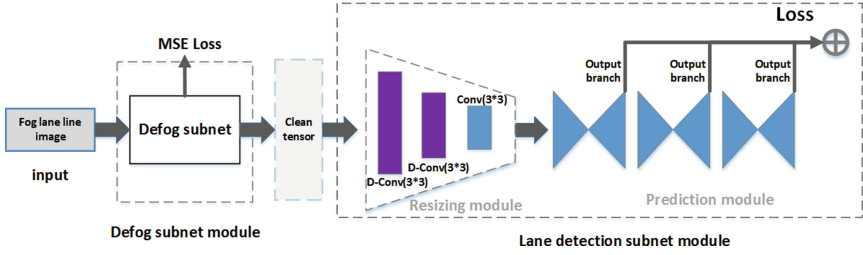


Fig. 1. Dual-subnet model: First, the foggy lane line image was processed through the defog subnet to obtain the clean feature map tensor with the same size as the input, which was resized to a $64 \times 32 \times 128$ feature map by resizing the network, and then the adjusted feature map was input to the prediction network for lane line detection.

accurate fog features and effectively improve the dehazing effect, we introduce the channel attention mechanism SE block into the defogging subnet. The channel attention mechanism introduced by us can greatly improve the network performance, although it will increase the computational consumption by a small amount. Moreover, the AOD-Net model is a lightweight network, so it can improve the subnet haze removal effect under the condition of ensuring the model prediction time. The input original image is added to the connection layer of the network as a residual block, and finally, the feature tensor of the clean image is obtained and input into the detection subnet. Table 1 shows the network details of the dehazing subnet.

In the specific detection process, the foggy RGB lane line image of size $512 \times 256 \times 3$ is converted into three matrices of size 512×256 as the input of the dual-subnet model, where the numbers in the matrix represent the pixels in the image. In the defogging subnet, the $512 \times 256 \times 3$ matrix is input into “conv1” with a convolution kernel size of 1×1 , and the result of “conv1” is input into “SeLayer1” (channel attention layer 1). Then, we input the result of “SeLayer1” to “conv2” with a convolution kernel size of 3×3 and then input the result of “conv2” to “SeLayer2”. The “concat1” concatenates features from the “SeLayer1,” “SeLayer2” and input image. The result of “concat1” is used as the input of “conv3”, and then the calculation continues. Similarly, “concat2” concatenates those from “SeLayer2”, “SeLayer3” and the input image; “concat3” concatenates those from “SeLayer1”, “SeLayer2”, “SeLayer3”, “SeLayer4” and the input image. After 5 convolutional layers, the clean tensor generation module finally retains the dehazed lane line feature map (clean tensor) with a size of $512 \times 256 \times 3$ as the input of the lane line detection module.

3.2 Resizing Module

As shown in Fig. 1, the resizing network is contained in the detection module behind the fogging subnet. To save memory and prediction time, the network is adjusted to reduce the size of the input image tensor. To make the output denser, it is suitable for instance segmentation tasks, and we increase the receptive field without reducing the resolution. We transform the first and second layers of the original resizing network into dilated

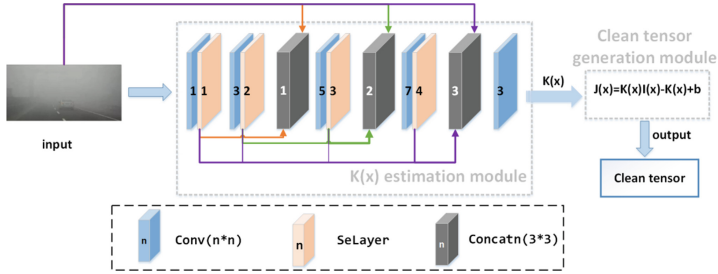


Fig. 2. Improved AOD-Net

Table 1. Details of the defogging subnet

Layer	Size/Stride	Output size
Input data		512*265*3
Conv1	1/1	512*265*3
SeLayer1		512*265*3
Conv2	3/1	512*265*3
SeLayer2		512*265*3
Concat1		512*265*9
Conv3	5/1	512*265*3
SeLayer3		512*265*3
Concat2		512*265*9
Conv4	7/1	512*265*3
SeLayer4		512*265*3
Concat3		512*265*15
Conv5 (K)	3/1	512*265*3

convolution operations with dilated rates of 3 and 2, respectively. Table 2 shows the details of tuning the network component layer.

Before detection, the lane line feature maps of size $512 \times 256 \times 3$ obtained by the defogging subnet are input into the resizing network. In the first layer of the resizing network, 32 convolution kernels with a size of 3×3 and dilated rate of 3 are used, and feature maps with a size of $256 \times 128 \times 32$ are obtained through the convolution of the first layer. In the second layer, 64 convolution kernels with a size of 3×3 and dilated rate of 2 are used, and feature maps of $128 \times 64 \times 64$ are obtained through the second layer convolution. In the third layer, 128 standard convolution kernels with a size of 3×3 are used, and feature maps of $128 \times 64 \times 64$ are obtained through the third layer convolution. Finally, $128 \times 64 \times 32$ feature maps were input into the stacked hourglass network.

Table 2. Details of resizing the network

Layer	Size/Stride/Padding	Dilation	Output size
Input data			512*265*3
Conv + PReLU + BN	3/2/2	3	256*128*32
Conv + PReLU + BN	3/2/1	2	128*64*64
Conv + PReLU + BN	3/2/1		64*32*128

3.3 Prediction Module

The PINet [35] lane line detection model uses a deep learning model inspired by stacked hourglass networks to predict some key points on a lane line. The model transforms the clustering problem of predicting key points into an instance segmentation problem to generate points on the lanes and discriminate the predicted points into individual instances. The network for extracting lane line features and making detections in this model consists of four stacked hourglass networks, so the network size can be adjusted according to the computing power of the target system (cutting several hourglass modules) without changing the network structure or performing extra training. The original PINet model is finished by stacking 4 hourglass blocks, with three output branches applied at the end of each hourglass block. They are the prediction confidence, offset, and embedding features, respectively. The loss function can be calculated based on the output of each hourglass block.

This stacked network model can transfer information to deeper layers, thus improving detection accuracy. Therefore, with knowledge distillation, we can expect better performance in cropped networks. However, with the increase in the number of stacked hourglass models, the number of parameters of the whole detection model also increases, and the detection speed is reduced. To balance high detection accuracy and high detection speed, our model stacks three hourglass blocks. The basic structure of the hourglass model is shown in Fig. 3. Some jump connections transfer information of different scales to deeper layers, and each colour block is a bottleneck module. These bottleneck modules are shown in Fig. 4. There are three types of bottlenecks: the same bottleneck, the down bottleneck, and the up bottleneck. The same bottleneck produces output of the same size as the input. The first layer of the “down bottleneck” is replaced by a dilated convolutional layer with a filter size of 3, a stride of 2, a padding of 2, and a dilated rate of 2. Add a channel attention mechanism after each “down bottleneck”. Table 3 shows the detailed information about the detection subnet.

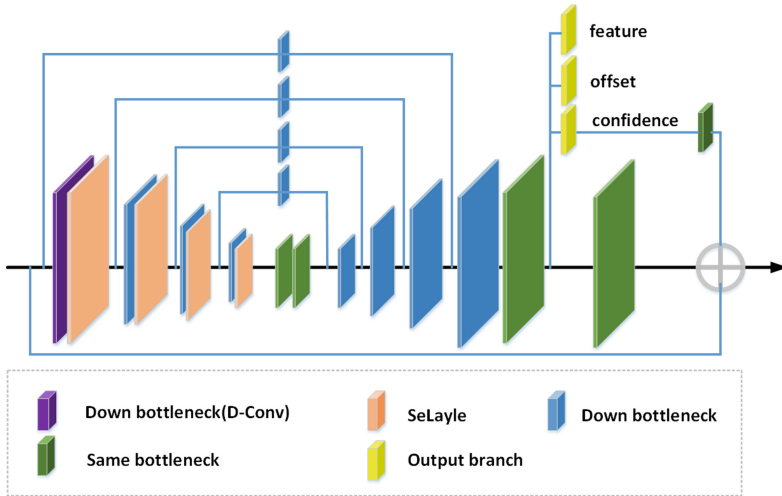


Fig. 3. Basic structure of an hourglass block

During detection, 128 feature map tensors with a size of 64×32 output from the resizing network are input to the stacked hourglass model. First, the feature map tensor with a size of $4 \times 2 \times 128$ is obtained through four downsampling layers, and then the feature map with a size of $64 \times 32 \times 128$ is restored after four upsampling layers. Each output branch has three convolution layers and generates a 64×32 grid. Confidence values about the key point existence, offset, and embedding feature of each cell in the output grid are predicted by the output branches. The channel of each output branch is different (confidence: 1, offset: 2, embedding: 4), and the corresponding loss function is applied according to the goal of each output branch.

Table 3. Lane line detection subnet details

	Layer	Size/Stride	Output size
	Input data		$64 \times 32 \times 128$
Encoder	Bottleneck (down)		$32 \times 16 \times 128$
	SeLayer1		$32 \times 16 \times 128$
	Bottleneck (down)		$16 \times 8 \times 128$
	SeLayer2		$16 \times 8 \times 128$
	Bottleneck (down)		$8 \times 4 \times 128$
	SeLayer3		$8 \times 4 \times 128$
	Bottleneck (down)		$4 \times 2 \times 128$
	SeLayer4		$4 \times 2 \times 128$

(continued)

Table 3. (continued)

	Layer	Size/Stride	Output size
(Distillation layer)	Bottleneck		4*2*128
	Bottleneck		4*2*128
	Bottleneck		4*2*128
	Bottleneck		4*2*128
Decoder	Bottleneck (up)		8*4*128
	Bottleneck (up)		16*8*128
	Bottleneck (up)		32*16*128
	Bottleneck (up)		64*32*128
Output branch	Conv + PReLU + BN	3/1	64*32*64
	Conv + PReLU + BN	3/1	64*32*32
	Conv	1/1	64*32*C

3.4 Loss Function

The training of the entire network relies on the dehazing loss and detection loss. A simple mean squared error (MSE) loss function is used in the defog subnet model. The detection loss is the sum of the output losses of each hourglass block, and the output branch of each hourglass network block includes four loss functions. As shown in Table 3, three loss functions are applied separately to each cell of the output grid. Specifically, the output branch generates 64 grids, and each cell in the output grid consists of 7 channels of predicted values, including confidence values (1 channel), offset values (2 channels), and embedded feature values (4 channels). The confidence value determines whether the keypoint of the traffic line exists. The offset value locates the exact location of

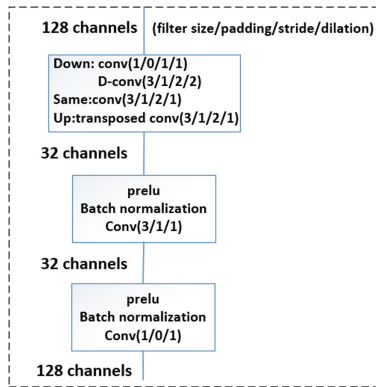


Fig. 4. Bottleneck details. The three kinds of bottlenecks have different first layers according to their purposes.

the keypoint predicted by the confidence value and utilizes the embedding feature to distinguish the key point as a single instance. The distillation loss function for extracting teacher network knowledge is adapted to the distillation layer of each encoder.

Dehazing Loss. MSE is the average value of the square of the difference between the predicted value and the true value calculated element by element. The calculation formula is shown in formula (1), where n is the total number of sample sets, \hat{y} is the predicted value of the i th sample, and y is the true value of the i th sample.

$$Loss(\hat{y}, y) = \frac{1}{n} \sum (\hat{y}_i - y_i)^2 \tag{1}$$

Confidence Loss. The confidence output branch predicts the confidence value for each cell. The confidence value is close to 1 if there is a key point in the cell and 0 otherwise. The output of the confidence branch has 1 channel and is fed into the next hourglass module. As shown in Eq. (2), the confidence loss consists of two parts, the presence loss and the absence loss. The presence loss is used for cells containing key points, and the absence loss is used to reduce the confidence value of each background cell. No loss is computed in cells with predicted confidence values above 0.01.

$$L_{exist} = \frac{1}{N_e} \sum_{C_C \in G_e} (c_c^* - c_c)^2, \\ L_{non-exist} = \frac{1}{N_n} \sum_{\substack{C_C \in G_e \\ c_c > 0.01}} (c_c^* - c_c)^2 + 0.00001 * \sum_{c_c \in G_n} c_c^2 \tag{2}$$

where N_e represents the number of cells containing key points, N_n represents the number of cells that do not contain any key points, G_e represents a set of cells containing key-points, G_n represents a set of cells containing points, and c_c represents the confidence level of the predicted value. For each cell in the output branch, c_c^* represents the true value.

Offset Loss. The offset branch predicts the exact location of the keypoint for each output cell. The output value for each cell is between 0 and 1, and the value indicates the position relative to the corresponding cell. As shown in Eq. (3), the offset branch has two channels for predicting the x-axis and y-axis offset.

$$L_{offset} = \frac{1}{N_e} \sum_{C_x \in G_e} (c_x^* - c_x)^2 + \frac{1}{N_e} \sum_{C_y \in G_e} (c_y^* - c_y)^2 \tag{3}$$

Embedding Feature Loss. When the embedding features are the same, the training branch makes the embedding features of each unit closer. Formula (4) is the loss function of the feature branch:

$$L_{feature} = \frac{1}{N_e^2} \sum_i^{N_e} \sum_j^{N_e} l(i, j), \\ l(i, j) = \begin{cases} \|F_i - F_j\|_2, & I_{ij} = 1 \\ \max(0, K - \|F_i - F_j\|_2), & I_{ij} = 0 \end{cases} \tag{4}$$

where F_i represents the predictive embedding feature of cell i , I_{ij} represents whether cell i and cell j are the same instances and K is a constant such that $K > 0$. If $I_{ij} = 1$, the cells are the same instance, and if $I_{ij} = 0$, the cells are different instances.

Distillation Loss. The more stacked hourglass modules there are, the better detection performance is, so the deep hourglass module can be the teacher network. After using the distillation learning method, the short network that is lighter than the teacher network will show better performance. The distillation loss is shown in Eq. (5).

$$L_{distillation} = \sum_m^M D(F(A_M) - F(A_m)),$$

$$F(A_M) = S(G(A_m)), S : \text{spatialsoftmax},$$

$$G(A_m) = \sum_{i=1}^C |A_{mi}|^2, G : R^{C \times H \times W} \rightarrow R^{H \times W}, \quad (5)$$

where D represents the sum of squares and A_m represents the distillation layer output of the m th hourglass module. M represents the number of hourglass modules, and A_{mi} represents the i th channel of A_m . Similar to the summation and exponential sum operations, the absolute value ($|\cdot|$) operations are calculated elementwise.

The total detection loss is the weighted sum of the above four loss terms, and the total loss is shown in Eq. (6).

$$L_{total} = \gamma_e L_{exist} + \gamma_n L_{non-exist} + \gamma_o L_{offset} + \gamma_f L_{feature} + \gamma_d L_{distillation} \quad (6)$$

γ_o is set to 0.2, γ_f to 0.5, and γ_d to 0.1. Both γ_e and γ_n are set to 1, with γ_e varying from 1.0 to 2.5 during the last 40 training periods.

4 Experiments

In this section, we first introduce the dataset details and the information for adding foggy lane lines to TuSimple [36] and CULane [26]. Second, we describe the evaluation indexes of the two datasets, the experimental environment and the implementation details. Finally, we show our experimental results and make some comparisons and analyses of the results.

4.1 Dataset

Most of the images in TuSimple were taken on sunny days with good lighting conditions and are often used for lane detection on structured roads. CULane is a large and challenging dataset that provides many challenging road detection scenarios. In this paper, a new class of foggy lane scenario types is added to TuSimple and CULane by using the fog synthesizing method of the atmospheric scattering model, and the fogged dataset is used to train and verify our model. To simulate the influence of different degrees of fog, the key parameter brightness A ranges from 0.68 to 0.69, and the fog concentration

parameter Beta ranges from 0.11 to 0.12. The original image is shown in Fig. 5 below, and the image after fogging by the above method is shown in Fig. 6. Table 4 shows a detailed data comparison of the two datasets before and after the fogging operation. Table 5 shows the comparison of the test set of the CULane dataset before and after the fogging operation.



Fig. 5. The original image



Fig. 6. Image after fogging

Table 4. Comparison of the datasets and the fogged datasets

Category	TuSimple	Fogged TuSimple	CULane	Fogged CULane
Frames	6,408	7,637	133,235	140,544
Train	3,268	4,036	88,880	96,864
Validation	358	358	9,675	9,675
Test	2,782	3,601	34,680	43,680
Resolution	1280 × 720	1280 × 720	1640 × 590	1640 × 590
Scenarios	1	2	9	10

4.2 Evaluation Indicators

Because different lane datasets have different collection devices and collection methods, as well as different lane line labelling methods, different indicators are generally used to evaluate the accuracy and speed of detection. This paper adopts the official TuSimple and CULane evaluation criteria.

(a) TuSimple

The main evaluation indexes of this dataset are accuracy, false positives and false negatives. The specific expressions are shown in (7), (8) and (9). Table 6 shows the meaning of each symbol.

$$Accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}} \tag{7}$$

Table 5. Comparison of the test sets of CULane and fogged CULane

Category	CULane	Fogged CULane
Normal	9,610	9,610
Crowded	8,115	8,115
Night	7,040	7,040
No line	4,058	4,058
Shadow	936	936
Arrow	900	900
Dazzle light	486	486
Curve	415	415
Crossroad	3,120	3,120
Fog	0	9,000

$$FalsePositive = \frac{F_{pred}}{N_{pred}} \quad (8)$$

$$FalseNegative = \frac{M_{pred}}{N_{gt}} \quad (9)$$

Table 6. The meaning of each symbol

Variable	Definition
C_{clip}	Number of correct points detected
S_{clip}	Number of true points
N_t	Number of lanes
F_{pred}	Number of lanes with errors detected
M_{pred}	Number of missed lanes
N_{gt}	Number of lanes with actual labels
N_{pred}	Actual number of lanes

(b) CULane

The CULane contains images of various road types, such as sheltered, crowded, urban, and nighttime roads. We followed the official evaluation criteria [B] to evaluate the CULane. According to [B], assuming that the width of each traffic line is 30 pixels, the intersection-over-union (IoU) ratio between the prediction of the evaluation model and the ground truth is calculated. Predictions above a certain threshold are considered

truly positive. Assuming that the threshold is set strictly at 0.5, the final score of the F1-measure is taken as the final evaluation indicator. The definition of the F1-measure is shown in (10).

$$F1_{measure} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (10)$$

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Recall = \frac{TP}{TP + FN} \quad (12)$$

TP denotes a true positive; that is, the prediction is true, and the reality is also true. FP denotes a false positive; that is, the prediction is true, but the reality is false. FN denotes a false negative; that is, the prediction is false, but the reality is true.

4.3 Experimental Environment and Implementation Detail

The experiments in this paper are performed in an Ubuntu16.04 operating system. The hardware configuration used in the experiment is as follows: CPU: Intel Core i9-10900K; GPU: NVIDIA RTX 3080. The programming language used is Python 3.7, and the deep learning development framework is PyTorch 1.8.

In the process of model training, the defogging subnetwork is first trained separately, then the weight of the defogging subnetwork is loaded and the partial network is frozen, and finally, the whole dual network is trained.

In the training process of the defogging subnet, the indoor NYU2 Depth Database processed by the atmospheric scattering model to synthesize fog in [10] is used, and the Gaussian random variable is used to initialize the weights. Using ReLU neurons, momentum and decay parameters are set to 0.9 and 0.0001, batch_size is set to 8, learning rate is set to 0.001, and training iteration period is set to 10. Then, when training the whole dual-subnet model, the weights trained by the defogging subnet are loaded into the dual-subnet model as pretrained weights, and this part of the defogging subnet is frozen. Then, the whole dual-subnet model is trained with fogged TuSimple and CULane. In the whole training process of the dual-subnet model, the confidence threshold was set to 0.36, the learning rate was set to 0.00001, the batch_size was set to 6, and the training iteration period was set to 1000. The distance threshold used to distinguish each instance is 0.08.

4.4 Experimental Results and Analysis

Fogged TuSimple. This paper requires exact X-axis and Y-axis values to test our dual-subnet model on the fogged TuSimple. The nH values in Tables 7 and 8 indicate that the network consists of n hourglass modules. Detailed evaluation results are shown in Table 7, and Fig. 7 shows the results on the images in the test set of the fogged TuSimple dataset. Table 7 summarizes the performance of our method, PINet(1H~4H), PolyLaneNet and SCNN(ResNet18, ResNet34, ResNet101) on the test set of fogged TuSimple. The first-

and second-best results are highlighted in red and blue, respectively, in Table 7. It can be seen from the results that the detection accuracy of the dual-subnet model with only three hourglass blocks is higher than that of the PINet model with four hourglass blocks. Therefore, the dual-subnet model achieves a good improvement in detection accuracy with the benefits of using a lighter model and calculating fewer parameters. By comparing the following detection algorithms, our method achieves better detection accuracy on the test set of foggy datasets, which also demonstrates the effectiveness of our defog subnet.

The last two columns of Table 7 show the number of frames set and parameters on the RTX 3080 GPU based on the number of hourglass modules. When only one hourglass block is used, the network detection speed is approximately 32 frames per second. When using four hourglass blocks, the dual-subnet model can run at 17 frames per second. Clipping a corresponding number of hourglass blocks can evaluate shorter networks without retraining. As the number of hourglass blocks increases, the dual-subnet model has higher performance and slower detection speed. The confidence thresholds are 0.35 (4H), 0.32 (3H), 0.30 (2H) and 0.52 (1H). By comparison, PolyLaneNet can output polynomials representing lane markers in images and obtain lane estimation values without post-processing, which can reach 115 FPS with fewer parameters. Although our method has a small number of parameters, it still needs to be defogged, so the detection speed is slow.

Table 7. Evaluation results on fogged TuSimple.

Method	Acc	FP	FN	fps	parameter(M)
PINet (1H)	89.42%	0.182	0.090	40	1.08
PINet (2H)	90.38%	0.168	0.084	35	2.08
PINet (3H)	91.75%	0.156	0.072	30	3.07
PINet (4H)	92.50%	0.150	0.080	25	4.06
PolyLaneNet[33]	91.16%	0.120	0.094	115	4.05
SCNN(ResNet18)[26]	90.67%	0.131	0.095	21	12.66
SCNN(ResNet34)[26]	91.14%	0.123	0.090	22	22.78
SCNN(ResNet101)[26]	91.82%	0.110	0.081	14	44.21
Ours (1H)	90.87%	0.171	0.092	32	1.20
Ours (2H)	91.01%	0.159	0.089	27	2.20
Ours (3H)	92.53%	0.147	0.072	22	3.19
Ours (4H)	92.54%	0.103	0.086	17	4.18

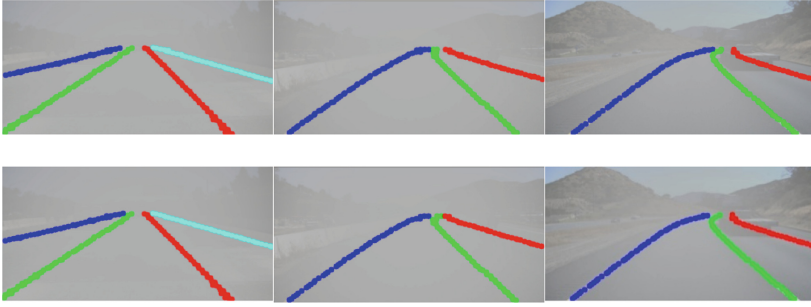


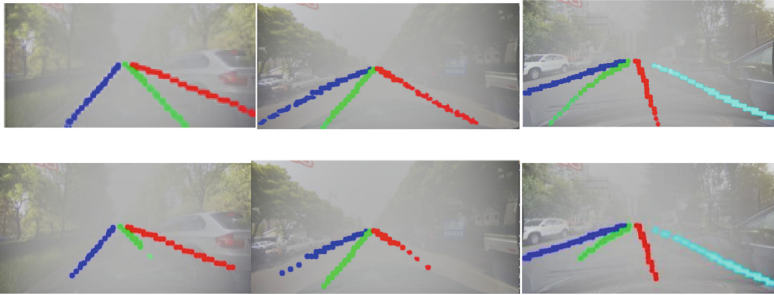
Fig. 7. Results on fogged TuSimple. The first row is the ground truth; the second row is the predicted results of the dual-subnet model.

Fogged CULane. Table 8 summarizes the performance of our method, PINet(1H~4H) and SCNN(ResNet18, ResNet34, ResNet101) on the test set of fogged CULane. The first- and second-best results are highlighted in red and blue, respectively, in Table 8. The following conclusions can be drawn from the experimental results in this paper. First, the detection accuracy of the dual-subnet model in a foggy environment is much higher than that of the original PINet model, and the detection effect in the dazzle light environment is also significantly improved. This is because our defog subnet has good de-fogging and de-noise effect in the fog and dazzling light environment, which makes the characteristics of the lane line more obvious. Second, in normal, no lane line, strong light and foggy environments, the dual-subnet model with three stacked hourglass blocks has higher detection accuracy than the original PINet model with four stacked hourglass blocks; that is, it realizes higher detection accuracy with a lighter model. However, the lane line detection subnets in the dual-subnet model in this paper are based on the key point estimation method, and local occlusion or unclear traffic lines will have a negative impact on performance. Therefore, the dual-subnet model performs poorly in crowded, arrow and curved environments. Finally, the dual-subnet model obtains a high F1 value on fogged CULane.

Figure 8 intuitively shows the detection effect of our dual-subnet model on the images in the test set of the fogged CULane dataset. Although the detection effect of our method is improved in the fogged environment, the detection effect of most methods is not good enough because vehicles block part of the lane lines in the figure.

Table 8. Evaluation results on fogged CULane.

Category	Normal	Crowed	Night	No Line	Shadow	Arrow	Dazzle Light	Curve	Crossroad(fp)	Fog
Proportion	22%	18.68%	16.12%	9.29%	2.14%	2.06%	1.11%	0.95%	7.14%	20.6%
PINet (1H)	82.17	61.22	48.97	36.60	59.01	75.31	56.10	57.41	1930	51.06
PINet (2H)	86.50	65.73	55.67	40.18	63.22	81.12	59.03	60.24	1892	52.97
PINet (3H)	87.47	66.45	55.89	40.30	64.67	80.55	59.45	60.01	1701	58.90
PINet (4H)	87.61	66.42	55.98	40.58	64.69	80.60	59.69	60.08	1622	59.81
SCNN(Res18)	88.03	65.39	56.10	39.69	62.15	78.18	58.50	58.21	1828	57.38
SCNN(Res34)	89.36	65.69	57.21	39.91	63.30	79.84	59.12	58.52	2178	58.69
SCNN(Res101)	90.01	66.41	58.09	40.20	65.01	81.76	60.22	60.21	1744	59.80
Ours (1H)	83.30	60.41	48.01	36.39	58.33	71.85	57.64	57.57	1869	56.21
Ours (2H)	87.21	64.51	54.29	40.12	62.76	76.23	59.20	60.78	1794	57.14
Ours (3H)	88.01	65.49	54.33	40.61	63.01	79.89	60.39	60.03	1622	63.01
Ours (4H)	88.03	65.76	54.77	40.70	63.85	78.97	61.03	60.15	1540	64.53

**Fig. 8.** Results on fogged CULane. The first row is the ground truth; the second row is the predicted results of the dual-subnet model.

5 Conclusion

In this paper, we propose a new dual-subnet model for lane detection in a foggy environment. The dual-subnet model combined with the defogging subnet module and the lane line detection module used the lightweight defogging subnet to improve the visual conditions in a foggy environment to improve the lane line detection accuracy. Our dual-subnet model achieves high accuracy and a low false positive rate and guarantees the safety of autonomous vehicles because the model rarely predicts the incorrect lane lines. Especially for detection in foggy environments, our dual-subnet model can improve the detection accuracy while ensuring fewer model parameters. However, in terms of detection speed, to realize visual enhancement, a preprocessing defogging process is added to our model, so the performance in terms of detection efficiency needs to be improved. In addition, in terms of datasets, a class of foggy lane line scenes was added to TuSimple and CULane by using a standard optical model, and these new datasets will be convenient for follow-up studies on lane line detection in foggy environments.

References

1. Narote, S.P., Bhujbal, P.N., Narote, A.S., Dhane, D.M.: A review of recent advances in lane detection and departure warning system. *Pattern Recognit.* **73**, 216–234 (2018)
2. Lv, C., Cao, D.P., Zhao, Y.F., et al.: Analysis of autopilot disengagements occurring during autonomous vehicle testing. *IEEE/CAA J. Automatica Sinica* **5**(1), 58–68 (2018). <https://doi.org/10.1109/JAS.2017.7510745>
3. Pei, S., Wang, S., Zhang, H., et al.: Methods for monitoring and controlling multi-rotor micro-UAVs position and orientation based on LabVIEW. *Int. J. Precis. Agric. Aviat.* **1**(1), 51–58 (2018)
4. Narote, S.P., Bhujbal, P.N., Narote, A.S., et al.: A review of recent advances in lane detection and departure warning system. *Pattern Recogn* **73**, 216–134 (2018). <https://doi.org/10.1016/j.patcog.2017.08.014>
5. Zhao, Z., Zhou, L., Zhu, Q.: Preview distance adaptive optimization for the path tracking control of unmanned vehicle. *J. Mech. Eng.* **54**(24), 166–173 (2018). (in Chinese)
6. Hillel, A.B., Lerner, R., Levi, D., et al.: Recent progress in road and lane detection: a survey. *Mach. Vis. Appl.* **25**(3), 727–745 (2014)
7. Zhang, X., Huang, H., Meng, W., et al.: Improved lane detection method based on convolutional neural network using self-attention distillation. *Sens. Mater.* **32**(12), 4505 (2020)
8. Chao, W., Huan, W., Chunxia, Z., et al.: Lane detection based on gradient-enhancing and inverse perspective mapping validation. *J. Harbin Eng. Univ.* **35**(9), 1156–1163 (2014)
9. Huang, S.C., Le, T.H., Jaw, D.W.: DSNet: joint semantic learning for object detection in inclement weather conditions. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**(8), 2623–2633 (2020)
10. Li, B., Peng, X., Wang, Z., et al.: An All-in-One Network for Dehazing and Beyond (2017)
11. Somawirata, I.K., Utamingrum, F.: Road detection based on the color space and cluster connecting. In: 2016 IEEE International Conference on Signal Image Process, pp. 118–122. IEEE (2016)
12. Qi, N., Yang, X., Li, C., Lu, R., He, C., Cao, L.: Unstructured road detection via combining the model-based and feature-based methods. *IET Intell. Transp. Syst.* **13**, 1533–1544 (2019)
13. Tapia-Espinoza, R., Torres-Torriti, M.: A comparison of gradient versus color and texture analysis for lane detection and tracking. In: 2009 6th Latin American Robotics Symposium, LARS 2009, pp. 1–6 (2009). <https://doi.org/10.1109/LARS.2009.5418326>
14. Küçükmanisa, A., Tarım, G., Urhan, O.: Real-time illumination and shadow invariant lane detection on mobile platform. *J. Real-Time Image Proc.* **16**(5), 1781–1794 (2017). <https://doi.org/10.1007/s11554-017-0687-2>
15. Wang, Y., Dahnoun, N., Achim, A.: A novel system for robust lane detection and tracking. *Signal Process.* **92**, 319–334 (2012). <https://doi.org/10.1016/j.sigpro.2011.07.019>
16. Aly, M.: Real time detection of lane markers in urban streets. In: IEEE Intelligent Vehicles Symposium Proceedings, pp. 7–12 (2008). <https://doi.org/10.1109/IVS.2008.4621152>
17. Mammeri, A., Boukerche, A., Lu, G.: Lane detection and tracking system based on the MSER algorithm, Hough transform and kalman filter. In: MSWiM 2014 - Proceedings of 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 259–266 (2014). <https://doi.org/10.1145/2641798.2641807>
18. Mu, C., Ma, X.: Lane detection based on object segmentation and piecewise fitting. *TELKOMNIKA Indones J. Electr. Eng.* **12**(5), 3491–3500 (2014)
19. Wu, P.-C., Chang, C.-Y., Lin, C.H.: Lane-mark extraction for automobiles under complex conditions. *Pattern Recognit.* **47**(8), 2756–2767 (2014)

20. Aung, T., Zaw, M.H.: Video based lane departure warning system using hough transform. In: International Conference on Advances in Engineering and Technology, pp. 29–30 (2014)
21. Marzougui, M., Alasiry, A., Kortli, Y., Baili, J.: A lane tracking method based on progressive probabilistic hough transform. *IEEE Access* **8**, 84893–84905 (2020). <https://doi.org/10.1109/ACCESS.2020.2991930>
22. Krizhevsky, I.S., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105 (2012)
23. Kim, J., Lee, M.: Robust lane detection based on convolutional neural network and random sample consensus. In: Loo, C.K., Yap, K.S., Wong, K.W., Teoh, A., Huang, K. (eds.) *ICONIP 2014*. LNCS, vol. 8834, pp. 454–461. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12637-1_57
24. Gurghian, T.K., Bailur, S.V., Carey, K.J., Murali, V.N.: Deeplanes: end-to-end lane position estimation using deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 38–45 (2016)
25. Zhang, W., Mahale, T.: End to end video segmentation for driving: lane detection for autonomous car, [arXiv:1812.05914](https://arxiv.org/abs/1812.05914) (2018)
26. Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: spatial CNN for traffic scene understanding. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (2018)
27. Ghafoorian, M., et al.: EL-GAN: Embedding Loss Driven Generative Adversarial Networks for Lane Detection (2019)
28. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer, [arXiv:1612.03928](https://arxiv.org/abs/1612.03928) (2016)
29. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network, [arXiv preprint arXiv:1503.02531](https://arxiv.org/abs/1503.02531) (2015)
30. Hou, Y., Ma, Z., Liu, C., Loy, C.C.: Learning lightweight lane detection CNNs by self-attention distillation. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1013–1021 (2019)
31. Yoo, S., Lee, H., Myeong, H., et al.: End-to-end lane marker detection via row-wise classification. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE (2020)
32. Qin, Z., Wang, H., Li, X.: Ultra fast structure-aware deep lane detection. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) *ECCV 2020*. LNCS, vol. 12369, pp. 276–291. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58586-0_17
33. Tabelini, L., Berriel, R., Paixao, T.M., Badue, C., De Souza, A.F., Oliveira-Santos, T.: PolyLaneNet: lane estimation via deep polynomial regression. In: *ICPR* (2020)
34. Feng, Z., Guo, S., Tan, X., et al.: Rethinking efficient lane detection via curve modelling. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17062–17070 (2022)
35. Ko, Y., Lee, Y., Azam, S., et al.: Key Points Estimation and Point Instance Segmentation Approach for Lane Detection (2020)
36. The tusimple lane challenge. <http://benchmark.tusimple.ai/>