Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 7544355, 12 pages
https://doi.org/10.1155/2021/7544355

Research Article

RS-Lane: A Robust Lane Detection Method Based on ResNeSt and Self-Attention Distillation for Challenging Traffic Situations

Ronghui Zhang, Yueying Wu, Wanting Gou, and Junzhou Chen

Guangdong Provincial Key Laboratory of Intelligent Transport System, School of Intelligent Systems Engineering, Sun Yat-Sen University, Guangzhou 510275, China

Correspondence should be addressed to Junzhou Chen; chenjunzhou@mail.sysu.edu.cn

Received 21 June 2021; Accepted 30 July 2021; Published 10 August 2021

Academic Editor: Xinqiang Chen

Copyright © 2021 Ronghui Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Lane detection plays an essential part in advanced driver-assistance systems and autonomous driving systems. However, lane detection is affected by many factors such as some challenging traffic situations. Multilane detection is also very important. To solve these problems, we proposed a lane detection method based on instance segmentation, named RS-Lane. This method is based on LaneNet and uses Split Attention proposed by ResNeSt to improve the feature representation on slender and sparse annotations like lane markings. We also use Self-Attention Distillation to enhance the feature representation capabilities of the network without adding inference time. RS-Lane can detect lanes without number limits. The tests on TuSimple and CULane datasets show that RS-Lane has achieved comparable results with SOTA and has improved in challenging traffic situations such as no line, dazzle light, and shadow. This research provides a reference for the application of lane detection in autonomous driving and advanced driver-assistance systems.

1. Introduction

Lane detection plays a vital role in autonomous driving. Reliable lane detection can help autonomous driving systems to make the right decisions. Lane detection algorithms get to be a challenging task due to many factors such as the wide variety of lane markings, the complex and changeable road conditions, and the inherent slender features of lane markings. In this paper, we proposed a lane detection method based on LaneNet [1] using Split Attention proposed by ResNeSt [2] and Self-Attention Distillation (SAD) [3] to improve the feature representation on the slender and sparse annotations like lane markings.

The current lane detection methods can be roughly divided into two kinds: one is based on traditional computer vision and the other one is based on deep learning. Most of the traditional detection methods rely on extracting a certain feature to detect lanes such as color features [4–6], edge features [7, 8], geometric features [9–11], and so on. Also, they are possibly combined with Hough Transform [12] and Random Sample Consensus (RANSAC) [13, 14]. These methods are simple and efficient, but they need to manually adjust the parameters. Although they can perform well when working in normal situations, they cannot adapt to situations with different conditions such as lighting and occlusion.

Most deep learning methods are based on Convolutional Neural Network (CNN) [1, 15, 16]. With the development of CNN, more and more theories and structures have been proposed. These advanced theories are also used in lane detection, such as attention mechanism [3], atrous convolution [17], semantic segmentation [17–19], instance segmentation [20, 21], and so on. Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) [22] are also used to detect lanes in continuous frames.

Although these methods have good performance in most normal situations, there are still some limitations. First of all, most methods can only detect a fixed number of lanes. Secondly, due to the slenderness of the lane, the number of background pixels is far greater than the number of lane

pixels. Learning such features could be very difficult. Also, some situations have few visual clues or even no clue at all, such as no line, shadow occlusion, and complex lighting conditions. Under these situations, detecting the lanes from the picture could be very tricky.

To solve the above problem, we use the framework proposed by LaneNet [1], using pixel embedding [23], to achieve instance segmentation so that our method can detect lanes without number limits. We use the Self-Attention Distillation (SAD) [3] mechanism and Split Attention proposed by ResNeSt [2] to improve the feature representation on the slender and sparse annotations like lane markings. Meanwhile, SAD would not increase the inference time of the model. The results on TuSimple dataset and CULane dataset show that our method is comparable with existing methods in normal situations. Above all, our method shows better improvement in situations with few visual clues, such as no line, shadow, and dazzle light.

Our contributions are summarized as follows:

(i) We use Split Attention to improve the feature representation of the network on the slender and sparse annotations like lane markings.

(ii) We use the distillation and attention mechanism of SAD to further improve the ability of the network without needing more annotation information. At the same time, it does not increase the inference time during deployment.

(iii) By using pixel embedding to obtain lane instances, our method can detect lanes without number limits, which is the limitation of most lane detection methods. Our method also shows great improvement in challenging traffic situations, such as no line, shadow, and dazzle light.

The rest of this paper is organized as follows. Section 2 reviews some related research of lane detection. Section 3 introduces the proposed lane detection method. Section 4 discusses our experimental results. Section 5 makes a conclusion of this paper.

2. Related Research

Vision-based lane detection methods can be divided into traditional vision methods and deep learning methods. However, both methods can be divided into three steps: image preprocessing, feature detection, and lane model fitting. Image preprocessing is to remove part of the noise. Feature detection uses the features of lanes to extract areas that are lanes. Lane model is fitted generally by using the least-squares method, spline fit, etc.

Among them, feature detection is the most important part of the lane detection algorithm and plays a decisive role in performance. There are many kinds of lane markings, including yellow lines, white lines, solid lines, and dashed lines. Moreover, the proportion of lane markings in the pictures is very low. And there may be abrasion and occlusion, which make detection more difficult.

Since AlexNet [24], Convolutional Neural Network (CNN) has been widely used in the field of computer vision with its outstanding feature extraction capabilities. More and more excellent neural networks have been proposed, such as ZFNet [25], GoogLeNet [26], VGGNet [27], and ResNet [28]. Since ResNet was proposed, it has been widely used as the backbone network due to its simple and modular structure. In recent years, there were many variants proposed based on ResNet, such as ResNeSt [2] and ResNeXt [29]. These networks are also widely used in the field of lane detection.

Semantic segmentation methods [30–32] are used to distinguish background and lane pixels. Instance segmentation methods [23] are used to directly get lane location. Object detection methods [33] are used to remove noise caused by cars and pedestrians.

In [1], they cast the lane detection problem as an instance segmentation problem. They designed a two branched, multitask network, for lane instance segmentation. In [3], a novel knowledge distillation approach is proposed, Self-Attention Distillation (SAD), which allows a model to learn from itself without any additional supervision or labels. Pan et al. [20] proposed a message passing mechanism between adjacent pixels to use visual information more efficiently, which significantly improves the performance of deep segmentation methods. In [21], a formulation with structural loss is proposed to address the problem of speed and no visual clue. The proposed formulation regards lane detection as a problem of row-based selection using global features and achieves remarkable speed and accuracy. In [22], a hybrid network combining CNN and RNN is proposed for robust lane detection in continuous driving scenes. In this framework, features on each frame of the input were firstly abstracted by a CNN encoder. Then, the sequential encoded features of all input frames were processed by a ConvLSTM. In [34], YOLO [33] and CPN [35] are used to remove noises, and then they proposed a lane marking model inference method that can detect lanes when lane markings are missing. But it can only detect two lanes. In fact, [3, 20–22, 34] are all limited with the fixed number of lanes.

In this paper, we design a network, which simultaneously performs semantic segmentation and instance segmentation and has no limits with the number of lanes. Our method also has better performance in challenging traffic situations.

3. Proposed Method

Our method can be divided into several modules as follows. In the preprocessing module, the input images can be appropriately processed to be easier to extract features in the later stage. The driving picture and its annotation are converted into a standard format that can be input into the model. In the model training stage, the annotated data are used to train the network so that it can achieve the segmentation of the lanes. The postprocessing stage is to get the final results from the output of the model through denoising and fitting. Figure 1 shows the flowchart of our method.

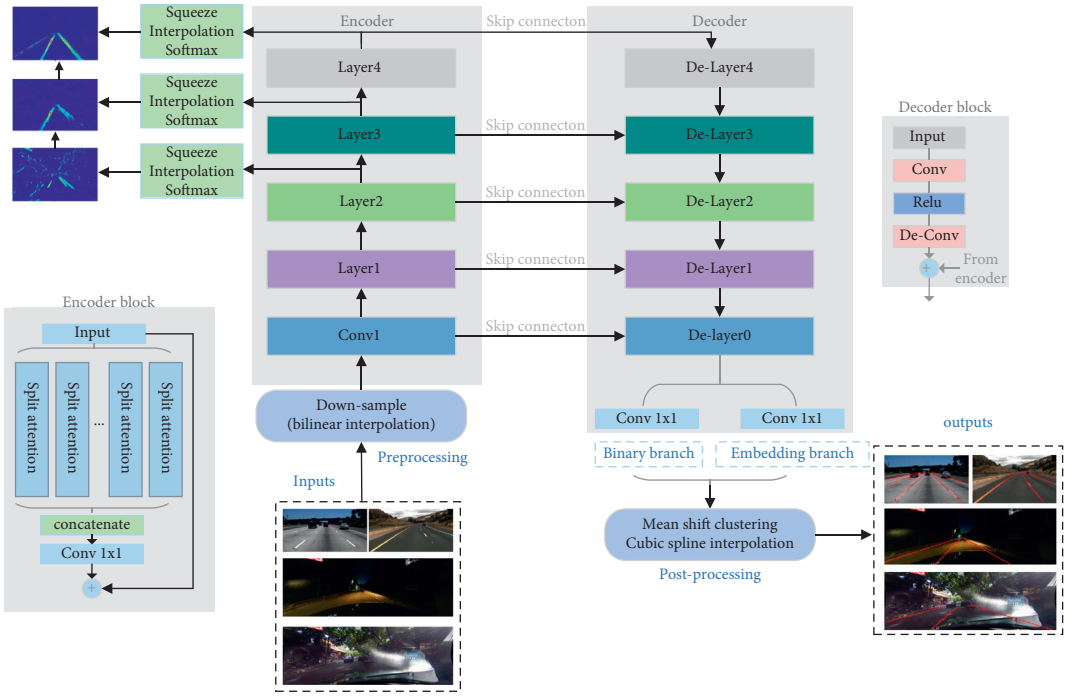


Figure 1: The flowchart of our method. Our network has one encoder, one decoder, and two branches. Through this network, images are segmented at the pixel level and the instance of each lane is obtained. Then, the cubic spline interpolation is used to fit each lane.

3.1. Preprocessing. This module processes the input image to make feature extraction easier in the later stage and improves the speed and accuracy. In our method, the preprocessing module is mainly to get the input image downsampled.

The original size of the images is 1280×720 in TuSimple dataset and 1960×590 in CULane dataset. If it is directly inputted into the network, the calculation is very large. Therefore, the raw image needs to be downsampled. In this paper, we use bilinear interpolation to downsample the images to the size of 512×288 (for TuSimple dataset) and 800×288 (for CULane dataset).

In our method, there are two branches in our network, which means two outputs. So, we made two labels for one input based on the annotations. One label is for binary branch, denoting whether a pixel belongs to lanes or background. And another one is for embedding branch, denoting which lane the pixel belongs to. Moreover, since the original image was downsampled, the same operation should be applied on the labels; the results are shown in Figure 2.

3.2. Model Training. Our network uses the structure proposed by LaneNet [1] to simultaneously perform semantic segmentation and instance segmentation, using the encoder-decoder [36] framework. Semantic segmentation is to achieve pixel-level processing of input pictures, getting the pixels that belong to lanes. Meanwhile, based on semantic segmentation, instance segmentation is performed using the pixel embedding [23] method proposed by De Brabandere et al.

Differently, LaneNet uses one encoder and two decoders, which means each branch has a decoder. To reduce the parameters and complexity of the network, we only have one decoder and use two different 1×1 convolution layers at the end of the decoder to get two branches. The binary branch is for semantic segmentation, and the embedding branch is for instance segmentation. The structure of the network is displayed in Figure 3.

3.2.1. The Encoder. Our encoder uses ResNeSt as the backbone. ResNeSt proposed a Split-Attention mechanism, which can obtain the attention based on different groups and different channels. As a variant of ResNet, ResNeSt retains the complete ResNet structure and has a conv1 and 4 layers. Each layer consists of several blocks. The structure of the encoder block is shown in Figure 4. Firstly, the encoder block divides the input into several groups (or cardinal) along the channel dimension. Then, each group is divided into several splits (or subgroups). After putting splits through different convolutions, the feature map of each group is a weighted combination of its splits, while the weights are selected based on contextual information. The output of the block is concatenated by feature maps of groups. This structure enables the network to utilize multidimensional information and enables cross-group and cross-channel attention.

We add two SAD paths to further enhance the feature extraction capabilities of the network. SAD enables a network to learn from itself, without needing any extract information. Through making the attention map of the lower layer to mimic the higher ones, the lower layers can learn the higher feature representation. Since the feature representation ability of lower layers is enhanced, the higher ones and the whole network also are enhanced.

The steps of applying SAD can be summarized as follows:

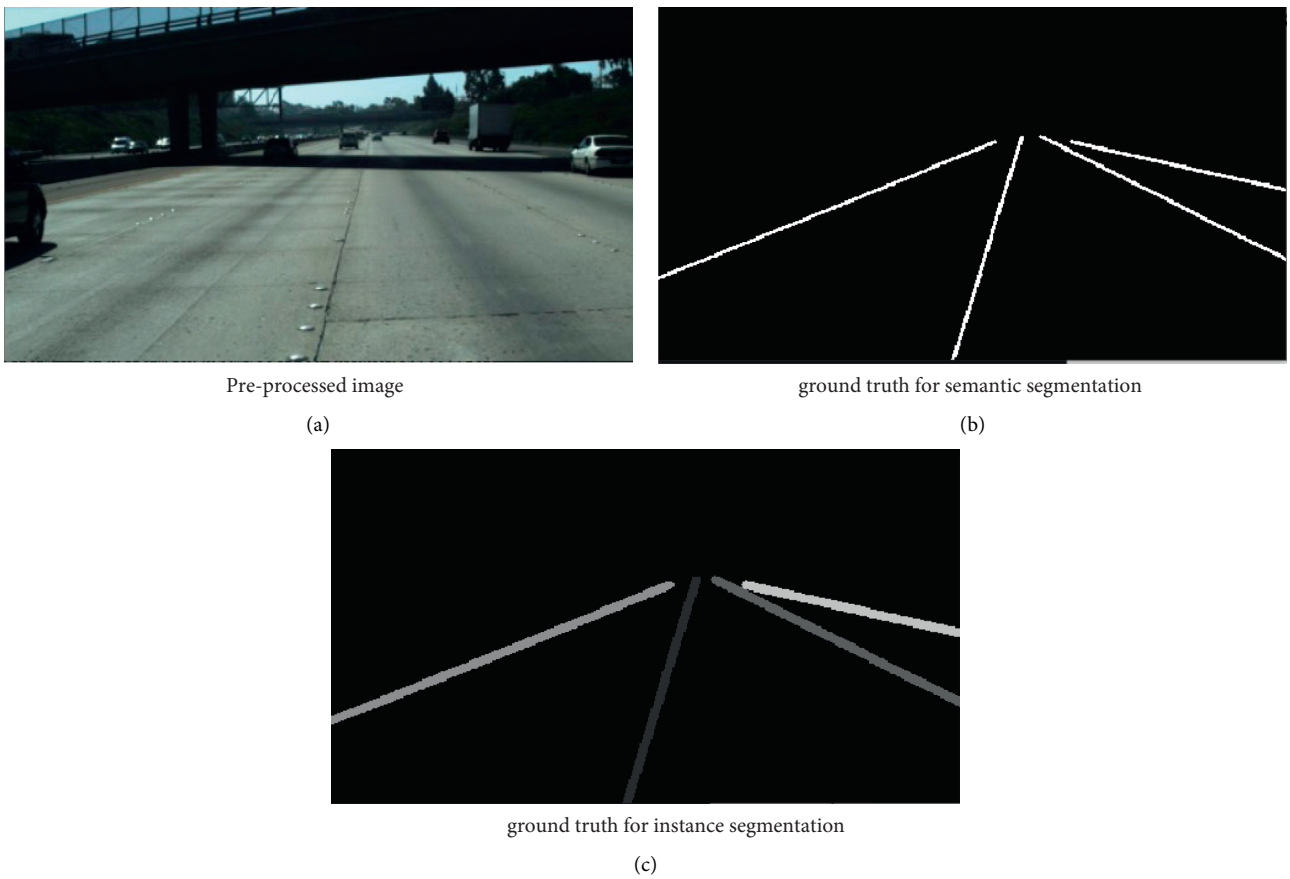


Figure 2: Ground truth. (a) Preprocessed image. (b) Ground truth for semantic segmentation. (c) Ground truth for instance segmentation.

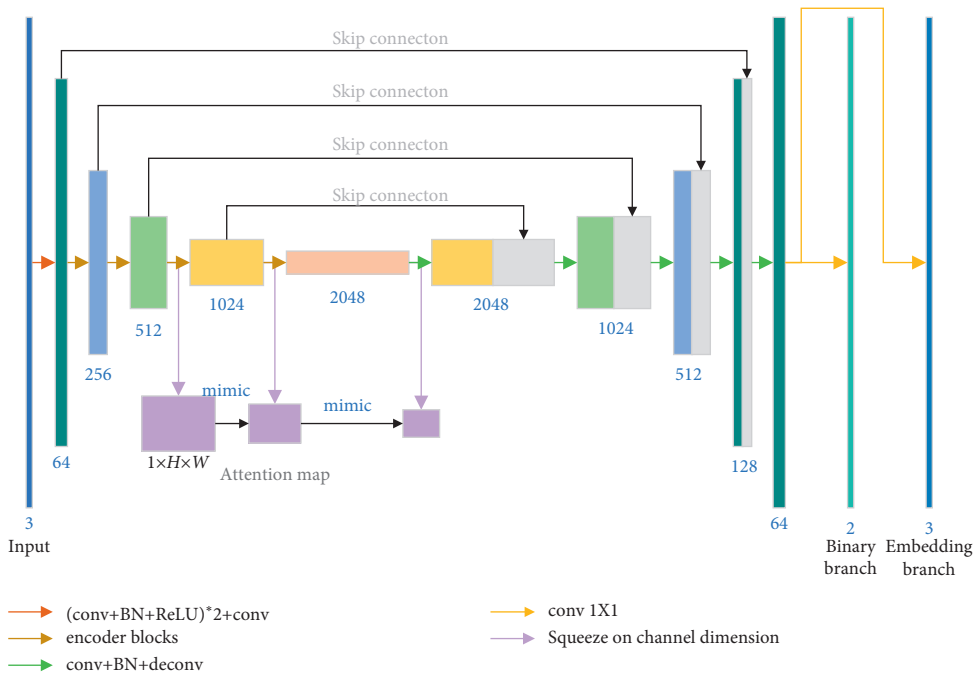


Figure 3: Network structure. Conv is a convolutional layer (convolution kernel is 3×3), deconv denotes the transposed convolution, BN denotes batch normalization, and Relu is the activation layer. Conv 1×1 is a convolutional layer with 1×1 kernel. Encoder blocks are shown in Figure 4. Blocks of the same color represent they have the same number of channels. Those gray ones mean they are from encoder.

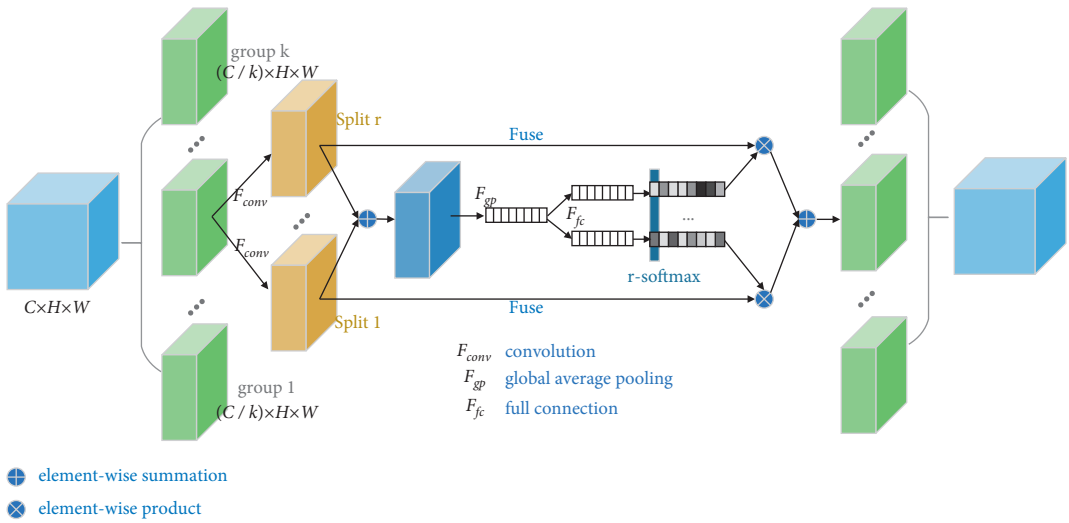


Figure 4: The structure of encoder block.

(i) *Squeeze*. The feature map is squeezed to a one-channel map, generating the attention map.

(ii) *Interpolation*. Upsample the size of target maps to mimic ones.

(iii) *Softmax*. Apply softmax operation on the same dimension for maps.

(iv) *Mimic Loss*. Get the MSE loss between mimic maps and target maps.

The first step is the generation of the attention map which is equivalent to finding a mapping function:

$$A_m \in R^{C_m \times H_m \times W_m}, \tag{1}$$

$$\mathcal{G}: \quad R^{C_m \times H_m \times W_m} \longrightarrow R^{H_m \times W_m}, \tag{2}$$

where A_m denotes the output of the m^{th} layer and C_m, H_m, and W_m, respectively, denote the channel, height, and width. And this mapping function can be constructed via computing statistics of these values across the channel dimension.

$$\mathcal{G}^p_{\text{sum}}(A_m) = \sum_{i=1}^{C_m} |A_{mi}|^p. \tag{3}$$

In our method, we set $p = 2$, following Hou et al. [3]. Additionally, Hou et al. [3] used the spatial softmax on the attention map. However, we find that, after using the spatial softmax, there are a large number of 0 existing in the map, and only several values are nearly 1, which is not conducive to the calculation value of the gradient. So, we apply softmax on the same dimension for maps.

3.2.2. The Decoder. The decoder performs deconvolution operation to decode the feature maps' output by the encoder and realizes upsampling and classification. Our decoder has 5 layers, one-to-one correspondence with layers of the encoder. In order to make full use of the global context information, we use skip-connect proposed by Unet [32] to concatenate the output of the encoder and the decoder. The structure of the decoder block is shown in Figure 1. In the last layer of the decoder, we have two branches, namely, binary branch and embedding branch. We use two convolutional layers with a 1×1 kernel to generate the output of binary branch and embedding branch. The binary branch outputs the semantic segmentation. The embedding branch outputs a three-channel embedding map, which means a 3D embedding vector for each pixel.

3.2.3. Loss Function. Loss function plays a great role in the optimization of the network. Our network has two outputs at the end, and the appropriate loss function needs to be selected.

Since the proportion of lane pixels in the image is very small, there is a serious data-imbalance problem in lane segmentation task. To solve this problem, we use dice loss as our semantic segmentation task loss.

After training, the embedding branch outputs a 3D embedding vector for each pixel. The distance between pixel embeddings belonging to the same lane is small. And the distance between pixel embeddings belonging to different lanes is maximized. De Brabandere et al. [23] introduced three terms to realize the loss function, variance loss (L_{var}), distance loss (L_{dist}), and regularization loss (L_{reg}). The variance loss (L_{var}) pulls the embedding of pixels to the mean embedding of a lane, that is, makes the embedding distance between pixels belonging to the same lane closer. The distance loss (L_{dist}) pushes the cluster centers apart from each other, that is, increases the embedding distance of pixels belonging to different lane lines. The function of the regularization loss (L_{reg}) is to attract all clusters to the origin.

Neven [1] et al. made some modifications to the loss function's formula and omitted L_{reg}. We use the loss function they modified, as shown in equations (4) and (5).

$$L_{\text{var}} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}\left[\left\|\mu_c - x_i\right\| - \delta_v\right]_+^2, \tag{4}$$

$$L_{\text{dist}} = \frac{1}{C(C-1)}\sum_{c_A=1}^{C}\sum_{c_B=1, c_A\neq c_B}^{C}\left[\delta_d - \left\|\mu_{c_A} - \mu_{c_B}\right\|\right]_+^2, \tag{5}$$

where δ_v and δ_d are hyperparameters, C denotes the number of clusters, N_c is the number of pixels belonging to cluster C, x_i is the embedding vector of the i^{th} pixel, μ_c is the mean embedding of cluster C, and $\|\cdot\|$ means the $L2$ distance.

We add two SAD paths in our network, which are between layer2, layer3, and layer4. After extracting the attention maps, since the target maps are smaller than origin maps, we upsample the target maps to the same size of the origin maps and perform softmax on each map. And then we calculate the Mean Square Error (MSE) between the mimic maps and the target maps. The SAD loss is formulated as follows:

$$L_{\text{SAD}}(A_m, A_{m+1}) = L_{\text{MSE}}(\Psi(A_m), \Psi(A_{m+1})), \tag{6}$$

where Ψ denotes the extraction, interpolation, and softmax operations.

The total loss include these three terms, as follows:

$$L_{\text{total}} = \alpha L_{\text{bin}} + \beta(L_{\text{var}} + L_{\text{dist}}) + \gamma L_{\text{SAD}}, \tag{7}$$

where L_{bin} is the loss of the semantic segmentation task calculated with the dice loss function. The parameters α, β, and γ balance the influence of each loss. In our experiments, we found the best performance when $\alpha = 1$, $\beta = 0.3$, and $\gamma = 0.1$.

3.3. Postprocessing. As mentioned above, there are two outputs of the network: one is the semantic segmentation map outputted by binary branch and the other is the embedding map outputted from the embedding branch. We use the segmentation map as a mask and apply the mask on the embedding map, so that we can get the embedding map only of lane pixels. Then we perform mean shift clustering on it to get clusters of each lane and obtain the real result of instance segmentation.

Most of the time, the beginning of lanes is very straight, and they start to bend in the end of the sight. Least square fitting cannot fit this kind of curve well. So, for each lane, we take the center points every 10 rows and get the final output through cubic spline interpolation.

4. Experiment Results

4.1. Development and Test Environment. We used Python as the main development language. The training and testing of the deep learning model are based on the PyTorch framework. The image processing part uses the OpenCV framework. The scientific computing part uses numpy and sklearn. The overall environment configuration parameters are shown in Table 1.

Table 1: Environment configuration parameters.

Name	Configuration
CPU	Intel® Core™ i7-10700K CPU @3.80 GHz
GPU	NVIDA GeForce RTX 2070 SUPER
CUDA version	10.1
Python version	3.7.9
Pytorch version	1.7.1(GPU)
OpenCV version	4.4.0

4.2. Dataset. We used two datasets to evaluate our method, TuSimple dataset [37] and CULane dataset [38]. TuSimple dataset collects road information at different times during the day, including two lanes/three lanes/four lanes/or more lanes, with different traffic conditions, including clear lane lines and severe wear. Some samples of the data set are shown in Figure 5.

CULane dataset is a much more challenging and larger dataset, including normal and 8 challenging situations, such as crowded, night, and no line. The proportion of different situations is shown in Figure 6. Some of the examples are shown in Figure 7.

Compared to CULane, TuSimple dataset is rather simple. But CULane dataset focuses on the detection of four lane markings, while TuSimple dataset includes four lanes or more. And the basic information about these two datasets is shown in Table 2.

4.3. Evaluation Metrics and Test Results. Usually, we regard lane detection as a binary classification problem, so the performance can be presented by confusion matrix. In addition, in order to compare with other methods, we also use the official evaluation metrics provided by datasets.

CULane dataset [38] uses precision, recall, and $F1$ to evaluate the detection. The expression of precision is given in equation (8), and the expression of recall is given in equation (9). TP (true positive) denotes the number of pixels that are lane pixels, and the predicted results are positive; FP (false positive) denotes the number of pixels that are not lane pixels, but the predicted results are positive; FN (false negative) denotes the number of pixels that are lane pixels, but the predicted results are negative. To measure these two values together, $F1$ is their harmonic mean, and the expression is shown in equation (10).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{8}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{9}$$

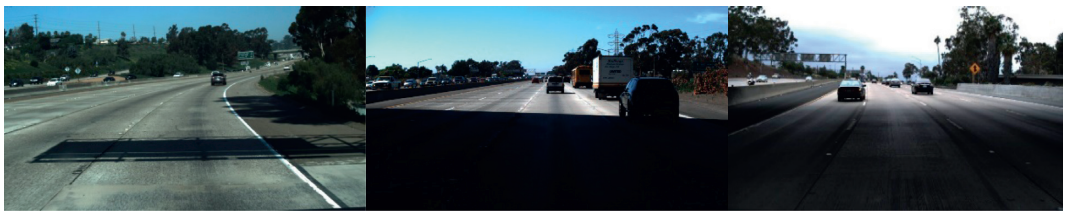
$$F1 = \frac{2}{(1/\text{precision}) + (1/\text{recall})}. \tag{10}$$

TuSimple [37] uses accuracy, which is computed as follows.

$$\text{Accuracy} = \frac{N_{\text{pred}}}{N_{\text{gt}}}, \tag{11}$$

Normal highway

(a)



Shadow

(b)

Figure 5: Some pictures in TuSimple dataset. (a) Normal highway. (b) Shadow.

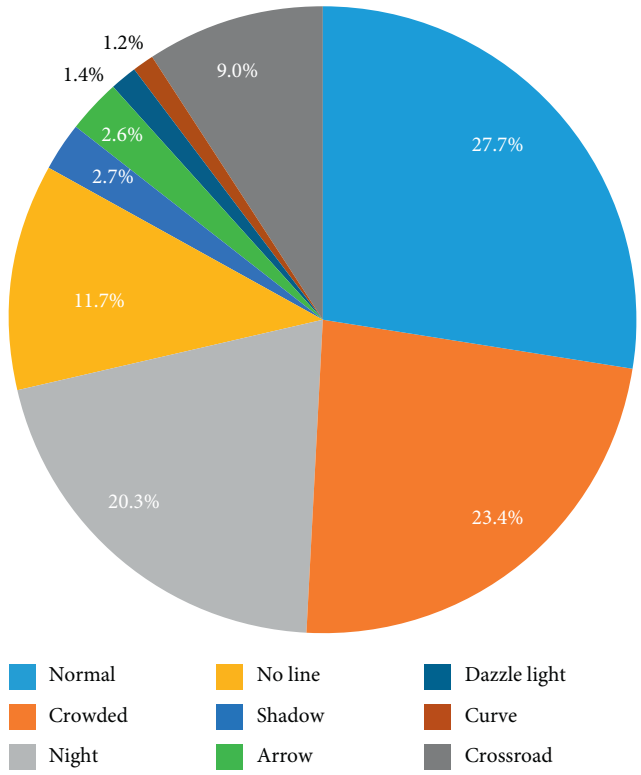
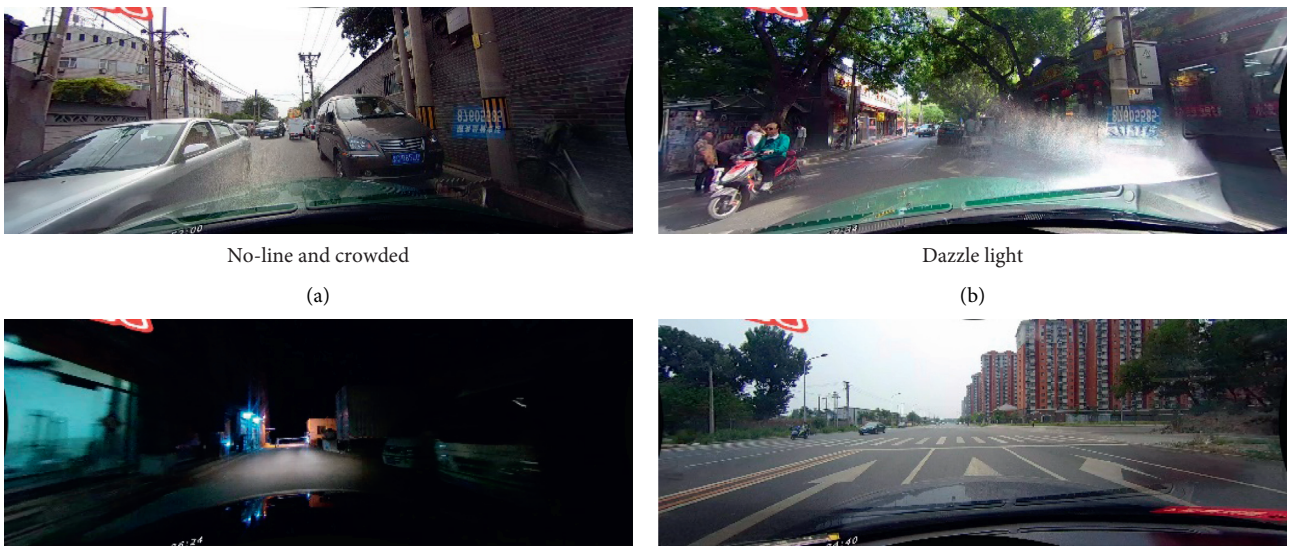


Figure 6: The proportion of different situations in CULane dataset [38].

where N_{pred} denotes the number of correctly predicted lane points and N_{gt} is the number of ground-truth lane points. Also, FN and FP are used as a reference.

The comparison of our method with SCNN, LaneNet, ENet-SAD, and EL-GAN on TuSimple dataset is given in Table 3. The comparison on CULane dataset is given in Table 4.

Table 3 shows our comparison results on TuSimple. Considering the accuracy of different methods, all are already extremely high, and the gap between our method and the best one is very small (which is 0.0027). It is fair to say that we have comparable performance with the state of the art on TuSimple. Table 4 shows that our method has better performance on challenging situations, especially on no line,



No-line and crowded (a) Dazzle light (b) Night (c) Arrow (d)

Figure 7: Some pictures in CULane dataset. (a) No line and crowded. (b) Dazzle light. (c) Night. (d) Arrow.

Table 2: Basic information about TuSimple dataset and CULane dataset.

Dataset	ImageNum	Size	Multilane	Time	Road type
TuSimple	6408	1280 * 720	No limits	Daytime	Highway
CULane	133,235	1640 * 590	Less than 4 lanes	Daytime and night	City, rural, highway

Table 3: Comparison on TuSimple testing set.

Algorithm	Accuracy	FN	FP
SCNN [20]	96.53%	0.0617	0.0180
LaneNet [1]	96.38%	0.0780	0.0244
FD50 [39]	95.20%	0.0760	0.0450
ENet-SAD [3]	96.64%	0.0602	0.0205
U-fast [21]	96.06%	—	—
EL-GAN [40]	96.39%	0.0412	0.0336
RS-Lane (ours)	96.37%	0.0532	0.0279

Accuracy is computed as equation (11). FN denotes the proportion of false negative points. FP denotes the proportion of false positive points.

shadow, and dazzle light situations. Overall our method has good performance in normal situations and also improved in challenging traffic situations.

The output of the trained network is shown in Figure 8.

From Figure 8, it can be seen that the trained network has a good recognition effect for lanes under various interferences, like the road shown in Figure 8(b) which is occluded by a large vehicle. The semantic segmentation stage fails to identify the occluded lane line pixels, but the instance segmentation stage can still classify the two lanes as the same lane.

The testing results on TuSimple dataset are shown in Figure 9.

Figure 9 shows part of the detection results of the lane line detection algorithm designed in this paper. It shows that our method has good performance on normal highway situations. And it can detect more than 4 lanes. When there is a large area of shadow on the road, our method can still work well.

The testing results on CULane dataset are shown in Figure 10. Since CULane contains more difficult situations, Figure 10 shows the performance of our method on crowded situations, hazzle light situations, and night and the performance when there are no line markings on the road or there are other markings.

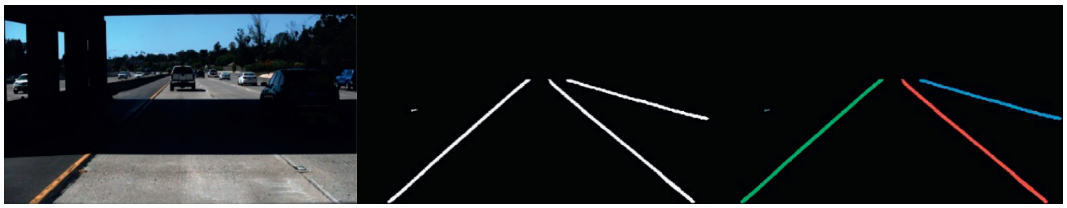
From the above testing results, it can be seen that our method has good robustness. It can deal with most of the normal conditions in the daytime and perform great in challenging conditions as well. The comparison also shows that our method has achieved the state of the art.

In order to measure the real-time performance of our method, we also test the running time of our method. The average running time of each step in our algorithm is shown in Table 5.

Table 4: Comparison on CULane testing set.

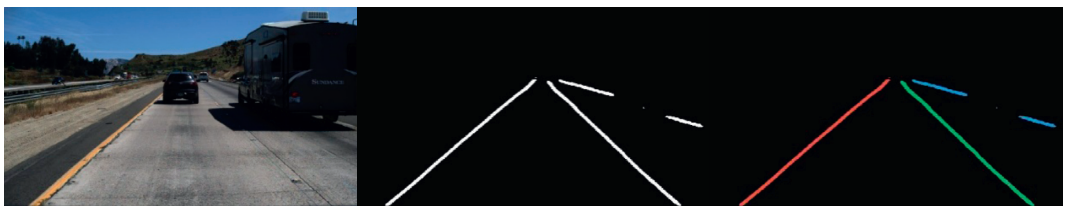
Category	SCNN [20]	FD50 [39]	U-fast [21]	ERFNet [41]	SAD [3]	Res101-SAD [3]	RS-Lane (ours)
Normal	90.6	85.9	90.7	**91.5**	90.1	90.2	88.5
Crowded	69.7	63.6	70.2	71.6	68.8	70	**72**
Dazzle light	58.5	57	59.5	66	60.2	59.9	**67.9**
Shadow	66.9	59.9	69.3	71.3	65.9	67	**75.7**
No line	43.4	40.6	44.4	45.1	41.6	43.5	**49.3**
Arrow	84.1	79.4	85.7	**87.2**	84	84.4	84.6
Curve	64.4	65.2	**69.5**	66.3	65.7	65.7	64.3
Crossroad	1990	7013	2037	2199	1998	2052	**1988**
Night	66.1	57.8	66.7	67.1	66	66.3	**68.3**
Total	71.6	—	72.3	73.1	70.8	71.8	**73.6**
Number of best	0	0	3	2	0	0	7

*F*1 values are shown for different categories in this table. FP values are shown for crossroad; since there is no line at crossroad, any prediction point will be false positive.



Shadow

(a)



Lane markings are severely worn and blocked by vehicles

(b)

Figure 8: Network outputs. The first column denotes inputs, the second column denotes semantic segmentations, and the third column denotes instance segmentations after clustering. (a) Shadow. (b) Lane markings are severely worn and blocked by vehicles.

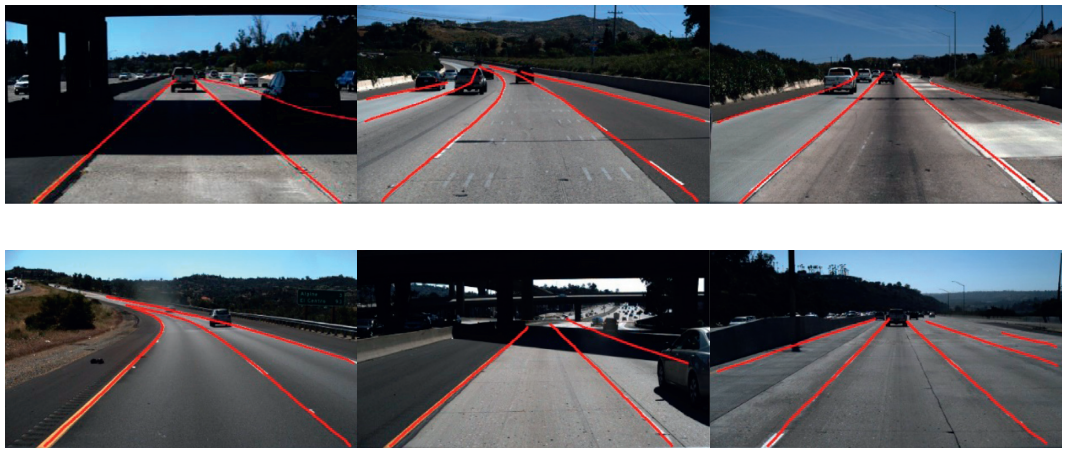


Figure 9: Testing results on TuSimple.

Figure 10: Testing on CULane. Since there is no line at crossroad, the detection result is none, as shown in (g). (a) Crowded. (b) Dazzle light. (c) Shadow. (d) No line. (e) Arrow. (f) Curve. (g) Crossroad. (h) Night.

Table 5: The running time of algorithm (test on TuSimple).

Process	Preprocessing and neural network	Clustering	Postprocessing	Total
Time	15.1 ms	31.3 ms	8.1 ms	54.5 ms

The real-time performance of the method on current hardware devices (shown in Table 1) is slightly off. The running time of our method now is about 54 ms per frame. Normally, 50 ms could satisfy the real-time requirements for most situations. The method mainly takes time on clustering algorithm, which takes up nearly half of the running time. This is because the cluster algorithm cannot speed up through GPU. But it can be made up by some simple solutions when applying to engineering. One is upgrading CPU and GPU, and the other is reducing the images' size when preprocessing. By doing these, our method can be applied in engineering.

5. Conclusion

In this paper, we present a new network for lane detection, which uses Split Attention and Self-Attention Distillation to enhance the performance in challenging traffic situations.

Using pixel embedding to obtain lane instances, our method also has no limits of number of lanes.

The results on TuSimple show that RS-Lane has comparable performance with the state of the art in most normal situations. And the results on CULane show that RS-Lane improves in challenging traffic situations such as no line and shadow. In general, our method achieves the state-of-the-art performance and provides a reference for the application of lane detection. Though the real-time performance is slightly off, our method can be applied in engineering by making a little change.

In the future, we will further explore how to improve the speed and accuracy at the same time. We will also continue working on the situations with various weather conditions, such as rainy and foggy. Besides, our work is a part of Cooperative Vehicle Infrastructure System, especially for future intelligent vehicle design and control with 5G technology [42–44].

Data Availability

The datasets are available from https://github.com/TuSimple/tusimple-benchmark and https://xingangpan.github.io/projects/CULane.html.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This study was partially supported by the National Key R&D Program of China (no. 2020YFB1600400), Guangzhou Science and Technology Plan Project (no. 202007050004), Shenzhen Fundamental Research Program (no. JCYJ20200109142217397), and Guangdong Natural Science Foundation (no. 2021A1515011794).

References

[1] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 286–291, Suzhou, China, June 2018.

[2] H. Zhang, C. Wu, Z. Zhang et al., "Resnest: split-attention networks," 2020, https://arxiv.org/abs/2004.08955.

[3] Y. Hou, M. Zheng, C. Liu, and C. Change Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, Seoul, Republic of Korea, November 2019.

[4] I. K. Somawirata and F. Utaminingrum, "Road detection based on the color space and cluster connecting," in *Proceedings of the 2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, Beijing, China, August 2016.

[5] K.-Y. Chiu and S.-F. Lin, "Lane detection using color-based segmentation," in *Proceedings of the IEEE Proceedings. Intelligent Vehicles Symposium 2005*, Las Vegas, NV, USA, June 2005.

[6] C. Ma, L. Mao, Y. Zhang, and M. Xie, "Lane detection using heuristic search methods based on color clustering," in *Proceedings of the 2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, Chengdu, China, July 2010.

[7] A. López, J. Serrat, C. Cañero, F. Lumbreras, and T. Graf, "Robust lane markings detection and road geometry computation," *International Journal of Automotive Technology*, vol. 11, no. 3, pp. 395–407, 2010.

[8] C. Hou, J. Hou, and C. Yu, "An efficient lane markings detection and tracking method based on vanishing point constraints," in *Proceedings of the 35th Control Conference*, pp. 6999–7004, IEEE, Chengdu, China, July 2016.

[9] Z. Teng, J.-H. Kim, and D.-J. Kang, "Real-time lane detection by using multiple cues," in *Proceedings of the ICCAS 2010*, Gyeonggi-do, Republic of Korea, October 2010.

[10] G. Liu, S. Li, and W. Liu, "Lane detection algorithm based on local feature extraction," in *Proceedings of the 2013 Chinese Automation Congress*, Changsha, China, November 2013.

[11] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, "A novel lane detection based on geometrical model and Gabor filter," in *Proceedings of the 2010 IEEE Intelligent Vehicles Symposium*, La Jolla, CA, USA, June 2010.

[12] A. Mammeri, A. Boukerche, and G. Lu, "Lane detection and tracking system based on the mser algorithm, Hough transform and Kalman filter," in *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Montreal, Canada, September 2014.

[13] J. Deng and Y. Han, "A real-time system of lane detection and tracking based on optimized ransac B-spline fitting," in *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, Montreal, Canada, October 2013.

[14] H. Tan, Y. Zhou, Y. Zhu, D. Yao, and K. Li, "A novel curve lane detection based on improved river flow and ransa," in *Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, China, October 2014.

[15] B. He, A. Rui, Y. Yang, and X. Lang, "Accurate and robust lane detection based on dual-view convolutional neutral network," in *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1041–1046, Gotenburg, Sweden, June 2016.

[16] Z. Wang, W. Ren, and Q. Qiu, "Lanenet: real-time lane detection networks for autonomous driving," 2018, https://arxiv.org/abs/1807.01726.

[17] Y. Sun, L. Wang, Y. Chen, and M. Liu, "Accurate lane detection with atrous convolution and spatial pyramid pooling for autonomous driving," in *Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dali, China, December 2019.

[18] W.-J. Yang, Y.-T. Cheng, and P.-C. Chung, "Improved lane detection with multilevel features in branch convolutional neural networks," *IEEE Access*, vol. 7, pp. 173148–173156, 2019.

[19] W. Van Gansbeke, B. De Brabandere, D. Neven, M. Proesmans, and L. Van Gool, "End-to-end lane detection through differentiable least-squares fitting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, Seoul, Republic of Korea, October 2019.

[20] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: spatial cnn for traffic scene understanding," 2017, https://arxiv.org/abs/1712.06080.

[21] Z. Qin, H. Wang, and X. Li, "Ultra fast structure-aware deep lane detection," 2020, https://arxiv.org/abs/2004.11757.

[22] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 41–54, 2019.

[23] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," 2017, https://arxiv.org/abs/1708.02551.

[24] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 2012.

[25] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Computer Vision-ECCV 2014*, vol. 8689, pp. 818–833, 2014.

[26] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016.

[29] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 2017.

[30] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: a deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[31] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: a deep neural network architecture for real-time semantic segmentation," 2016, https://arxiv.org/abs/1606.02147.

[32] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, Munich, Germany, October 2015.

[33] J. Redmon, S. Divvala, R. Girshick, and F. Ali, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016.

[34] Y. Ma, H. Vincent, J. Bai, and X. Zhu, "Vision-based lane detection and lane-marking model inference: a three-step deep learning approach," in *Proceedings of the 2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, Taipei, Taiwan, December 2018.

[35] C.-A. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler, "Convolutional patch networks with spatial prior for road detection and urban scene understanding," 2015, https://arxiv.org/abs/1502.06344.

[36] I. Sutskever, O. Vinyals, and V. L. Quoc, "Sequence to sequence learning with neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, Montreal, Canada, December 2014.

[37] "Tusimple benchmark," 2018, https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection.

[38] "Culane dataset," 2018, https://xingangpan.github.io/projects/CULane.html.

[39] J. Philion, "Fastdraw: addressing the long tail of lane detection by adapting a sequential prediction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June 2019.

[40] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann, "El-Gan: embedding loss driven generative adversarial networks for lane detection," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Munich, Germany, September 2018.

[41] T. Liu, Z. Chen, Y. Yang, Z. Wu, and H. Li, "Lane detection in low-light conditions using an efficient data enhancement: light conditions style transfer," in *Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA, June 2020.

[42] R.-H. Zhang, Z.-C. He, H.-W. Wang, F. You, and K.-N. Li, "Study on self-tuning tyre friction control for developing main-servo loop integrated chassis control system," *IEEE Access*, vol. 5, pp. 6649–6660, 2017.

[43] H. Xiong, Z. Tan, R. Zhang, Z. Zong, and Z. Luo, "Flexural behavior and mechanical model of aluminum alloy mortise-and-tenon *T*-joints for electric vehicle," *Nanotechnology Reviews*, vol. 8, no. 1, pp. 370–382, 2019.

[44] H. Hao, D. Hui, and D. Lau, "Material advancement in technological development for the 5G wireless communications," *Nanotechnology Reviews*, vol. 9, no. 1, pp. 683–699, 2020.