Hindawi
Security and Communication Networks
Volume 2021, Article ID 6618459, 13 pages
https://doi.org/10.1155/2021/6618459

Research Article

Lane Detection Based on Adaptive Network of Receptive Field

YuFan Cai, YanYan Zhang, and ChengSheng Pan

Department of Electronic and Information Engineering, Institute of Intelligent Network and Information System, Nanjing University of Information and Science Technology, Nanjing 210044, China

Correspondence should be addressed to YanYan Zhang; 002243@nuist.edu.cn

Received 17 November 2020; Revised 3 December 2020; Accepted 16 March 2021; Published 24 March 2021

Academic Editor: Mamoun Alazab

The difficulty of lane detection lies in the imbalance of the number of target pixels and background pixels. The sparse target distribution misleads the neural network to pay more attention to background segmentation in order to obtain a better loss convergence result. This makes it difficult for some models to detect lane line pixels and leads to the training fail (unable to output useful lane information). Increasing receptive field properly can enlarge the sphere of action between pixels, so as to restrain this trouble. Moreover, the interference information and noise existing in the real environment increase the difficulty of lane classification, such as vehicle occlusion, car glass reflection, and tree shadow. In this paper, we do think that the features obtained by the reasonable combination of receptive fields can help avoid oversegmentation of the image, so that most of the interference information can be filtered out. Based on this idea, Adaptive Receptive Field Net (ARFNet) is proposed to solve the problem of receptive field combination with the help of multireceptive field aggregation layers and scoring mechanism. This paper explains the working principle of ARFNet and analyzes several results of experiments, which are carried out to adjust network structure parameters in order to get better effects in the CuLane dataset testing.

1. Introduction

With the advent of 5G era, automatic driving as a new traffic concept has been developed rapidly. It requires the further development of lane detection technology based on computer vision, as it plays an important role in automatic driving technology. Current mainstream lane detection algorithms give up using single filter; rather, they choose methods of deep learning such as the convolution neural network or Transformer to transform the lane detection task into image segmentation. However, due to the special structural characteristics of lane lines, the number of distributions between background pixels and target pixels is seriously uneven. This sparsity seriously affects the quality of network results. In addition, the widespread interference information in the driving environment makes it difficult to distinguish lane line pixels from background pixels.

To solve these problems, Lee et al. [1] chose to engage more human interventions in the training process, using vanishing points to guide the training of deep models, but this way requires more manpower. Pan et al. [2] added a spatial information transfer module in VGG16 [3] network to effectively resist the sparsity effect, and the lane position can be located more accurately by transferring the local information to the global. However, this message passing process increases the inference time greatly. To avoid a large increase in time, Hou et al. [4] used self-attention distillation technology, which encourages the network to extract more global information from feature graph by adding distillation loss function. Lee et al. [5] combined the advantages of the above two methods to transfer information in the horizontal and vertical directions in the distillation diagram. The ESA module abandons the method of using the average of distillation diagram to learn layer by layer but directly learns from the label via MSE loss. Qin et al. [6] grasped the special mathematical distribution characteristics of lane pixels, taking advantage of the structure-aware method to detect lane lines faster and better. However, all methods of distillation still need additional loss function to constrain the learning direction, which means that much more human testing is needed to adjust the function form, and it will bring more challenges as well.

In addition to using Convolution Neural Network (CNN) and front view to model, Garnett et al. [7] not only input the bird's eye view into the CNN, but also took advantage of 3D modeling method to deal with the complex fork road and curve situation. Liu et al. [8] replaced CNN model with Transformer and greatly improved both the accuracy and speed. In fact, these two methods have high requirements for hardware configuration, so it is not easy to train the model.

In our work, without adding any additional loss function, ARFNet aggregates information from different receptive fields in the layer. It extracts the mean values of different feature graphs as global information, which will be scored in multilayer perceptron (MLP). This process does not add extra inference time, and at the same time, with the help of pruning [9], the network structure can be simplified without harming the performance according to the score. Moreover, we have done some detailed adaptive ability exploration experiments and achieved good results in lane detection.

2. Related Work

2.1. Network Structure Optimization Theory. CNN based deep learning models can be regarded as many feature streams constructed by multilayer filters, as shown in Figures 1 and 2.

If the convolution operations in layers are ignored, the forward propagation process can be described by a path matrix:

$$\begin{bmatrix} 1A & 1B & \cdots & OUT \\ 1A & 2B & \cdots & OUT \\ \vdots & \vdots & \vdots & \vdots \\ 1B & 2C & \cdots & OUT \end{bmatrix}. \tag{1}$$

The number of rows in the matrix represents the number of paths, and the number of columns represents the depth of the network. The elements in the matrix represent the feature maps. Each independent path will have different impacts on the classification judgment of different categories. Any modification of the matrix will affect the learning ability of the network. Therefore, the network design can be intervened by changing the number of matrix elements and changing the value of elements.

(1) Changing the number of matrix elements. EfficientNet [10, 11] is a good example. It uses a simple and efficient composite coefficient to enlarge the baseline from three dimensions: depth, width, and resolution. This makes the network structure more reasonable without any cost of manual adjustment. In other words, the size of the path matrix is optimized.

(2) Changing the value of elements. As shown in Figure 3, convolution process can be equivalent to matrix multiplication (Hadamard product). This means that each feature map can be described as shown in

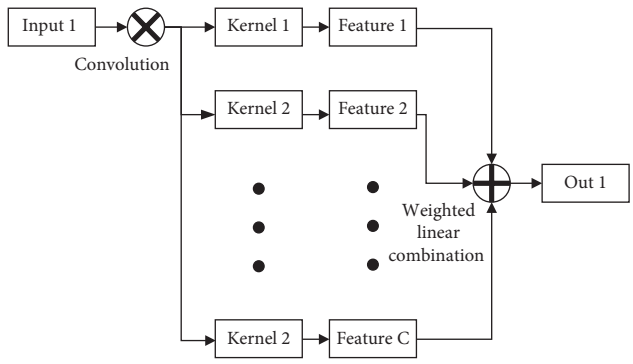


Figure 1: Working process in one convolution layer.

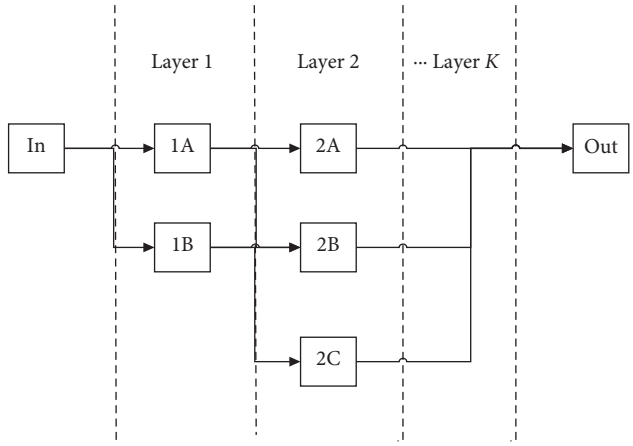


Figure 2: Feature streams in CNN.

$$\text{Feature} = \text{Data} * \prod_{l=1}^{L} W_l. \tag{2}$$

(3) Although the values of elements in kernel space are random and variable, we still can change the initialization distribution, like Xavier [12], or use the constraint mechanism to guide the change process. DyNet [13] uses covariance prediction module and dynamic convolution kernel mechanism to guide the weighted feature map forming to improve convolution efficiency. In addition, changing the distribution of the kernel space will affect the feature maps as well. DCN [14, 15] has adopted this strategy, which adds the offset to adjust the position of convolution kernel. Moreover, pruning [9] also can be combined to make the matrix more sparse, thus reducing the amount of calculation.

(4) Changing the initial data. Generally, the initial image will be rotated, clipped, or do value transformation to increase the generalization ability of the model, such as CutMix [16], PBA [17], and autoaugmentation [18]. In addition, Tsinghua team also found that the data augmentation can still be finished via loss function (ISDA [19]).

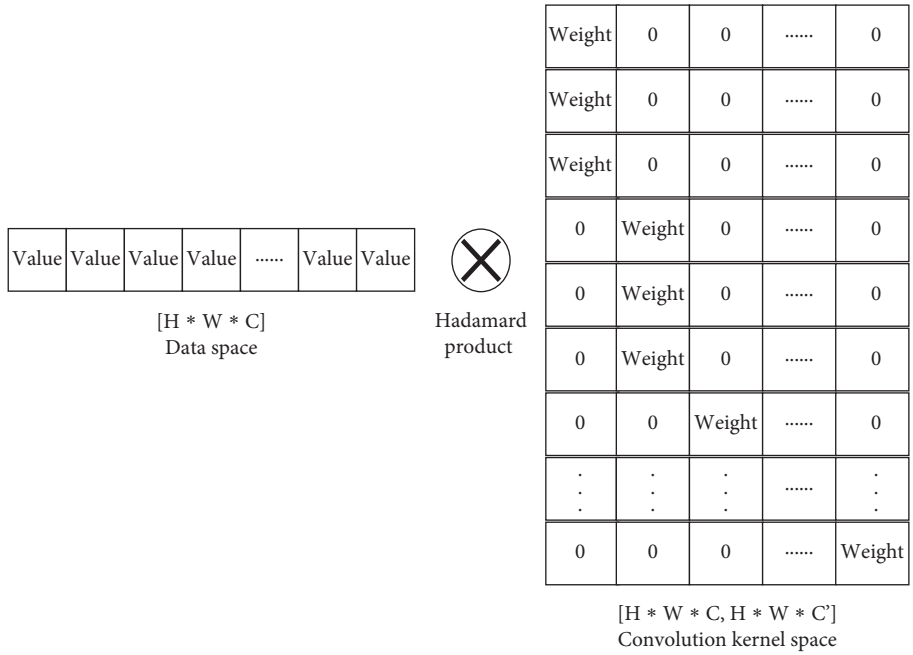


Figure 3: Equivalent expression.

In our work, due to the limitation of hardware equipment, it is necessary to limit the model size artificially. Therefore, instead of adopting the first option, we will integrate the second and third options.

2.2. ARFNet. In lane detection, appropriately increasing the receptive field can effectively improve the accuracy of classification. Inspired by the structure of InceptionNet [20], convolution modules with different receptive fields will be fused in parallel in one layer. Meanwhile, receptive fields can be easily changed by dilated convolutions [21]. Hence, in theory, the convolution module of ARFNet should be as shown in Figure 4.

By adding the squeeze-and-excitation-module [22, 23] (see Figure 5), We have achieved the scoring of information under different receptive fields. This effect will also be passed along with the accumulation of multiplication. Unfortunately, in the actual experiment, the number of parallel modules is limited; in other words, the dilation rate r is limited by the hardware storage capacity. Therefore, instead of adopting the strategy of structural search proposed in IC-conv [24] to implement self-adaption, our work chooses to manually adjust the dilation rate and use the scoring mechanism to let the network adapt to these features. At the same time, it is easier to find the effect of different receptive field aggregation on the results by manual adjustment. Specific details will be introduced in the experimental section.

3. Methodology

3.1. Loss Function. Lane detection is commonly formulated as a semantic segmentation task. Specifically, we assign the corresponding class number i ($i = 1, 2, 3, \ldots, N$) to the lane line pixels to facilitate the completion of a multiclassification task. At the same time, the lane existence label is added to assist network learning that uses the binary labels to describe the distribution of lanes. The total loss function is described as

$$\text{Loss} = L_{\text{seg}}(s, \hat{s}) + 0.5 * L_{\text{exist}}(b, \hat{b}), \tag{3}$$

where $(\hat{s}, \hat{b})$ is the predict map, and (s, b) is the label. In order to highlight the importance of the target in the background, in the cross-entropy loss function, the weight of the labels will be reassigned after one-hot processing. The weight distribution is arranged in the following order:

$$\text{Weight} = [\text{Background}, \text{Lane1}, \text{Lane2}, \text{Lane3}, \text{Lane4}]. \tag{4}$$

By actively suppressing attention to the background, the model can perform better under sparse labels. However, excessive restraint will bring more misjudgment that tends to distinguish more background pixels into lanes pixels. Based on this factor, after several testing and comparisons, we find that the weight distribution of [0.2, 2, 2.5, 2.5, 2] can bring better prediction results of lane pixels than just using the weight of [1, 1, 1, 1, 1]. Therefore, this paper will use the weight of [0.2, 2, 2.5, 2.5, 2] in the following experiments.

3.2. Dataset. Figure 6 shows a sample of the CuLane [25] dataset. Although the dataset contains only four lanes, its sample range is wide including a large number of complex scenes with background interference. It is an extremely challenging training dataset, so that it can well test the learning ability and generalization ability of the model. The specific dataset description is shown in Table 1 (the color changes in Figure 6(b) represent the occlusion, and interference has occurred in different degrees).

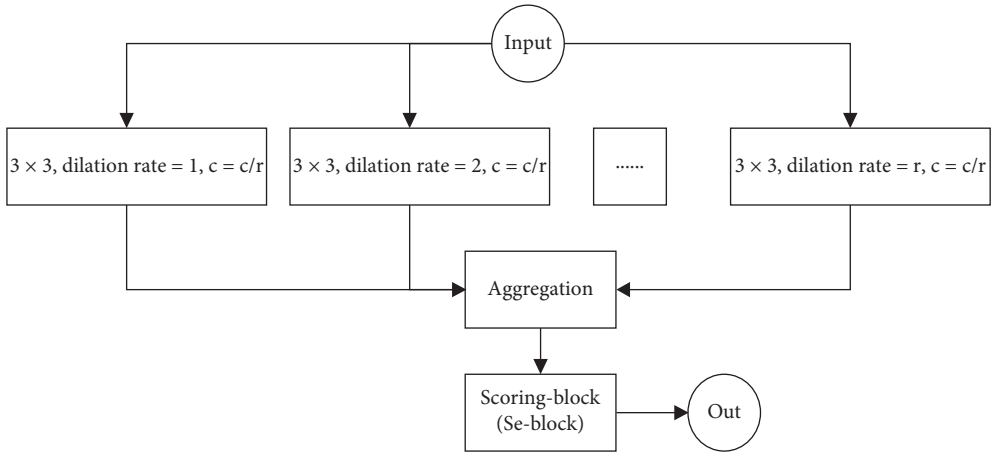


Figure 4: ARF-module.

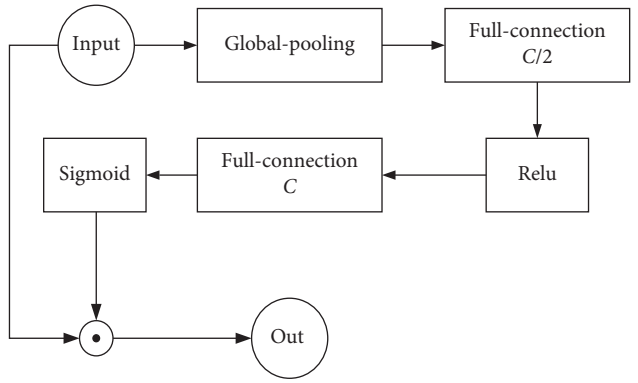


Figure 5: Se-module.

(a) (b)

Figure 6: (a) CuLane sample. (b) Overlay.

Table 1: Basic information of CuLane dataset.

Name	Train	Validation	Test	Resolution	Road type	Other label	Lane
CuLane	88880	9675	34680	1640 × 590	Urban, rural, highway	✓	4

3.3. Experiment Configuration and Strategy. In this paper, the only experimental GPU is RTX2080 super. Due to the limited video memory of 8G, we compress the original image to 800 × 216 and set the length of each batch to 10. In the experiment, we randomly selected 80 K images as the training set and 9 K images as the validation set. Moreover, we take advantage of SGD [26] algorithm and CosineAnnealing [27] algorithm to change the learning rate and optimize the parameter update. In the experiment, the initial learning rate is set to $1e-3$, the weight decay is set to $1e-4$, and the annealing period is set to 10 to enhance overfitting resistance. In addition, we retain the momentum part of 0.9

and allow the Nesterov [28] algorithm to correct the gradient direction. Because there are many comparison experiments in our experiments, we only normalize images as shown in equation (5) and will not use the dataset augmentation strategy or batch random processing in order to ensure the consistency of learning samples.

$$\text{data} = \frac{(\text{data} - \text{mean})}{\text{std}}. \tag{5}$$

Mean is the expectation of the data set, and std is the standard deviation of the data set. For the CuLane dataset, the parameter values are as

$$\begin{aligned} \text{mean} &= (0.3598, 0.3653, 0.3662), \\ \text{std} &= (0.2573, 0.2663, 0.2756). \end{aligned} \tag{6}$$

3.4. Evaluation Metrics. Different from the evaluation metrics in SCNN [2], which used IoU (see equation (7)) to increase fault tolerance and $F1$-score (see equation (8)) to evaluate the performance, we choose to use Accuracy (see equation (9)) and Recall (see equation (10)) of each lane line to comprehensively evaluate the performance.

$$F1 - \text{score} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{7}$$

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

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

$$\text{Accuracy} = \frac{N_{\text{pred}}}{N_{\text{gt}}}, \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{11}$$

where TP represents the number of lane pixels accurately classified, FN represents the number of lane pixels wrongly judged as background, FP represents the number of background pixels wrongly judged as lane pixels, N_{pred} represents the number of pixels correctly classified, and N_{gt} represents the total number of pixels. In the experiment, we find that the Precision is always much larger than the Recall. This is because the sparsity of the target leads to misjudge lane pixels as background pixels. This makes the Precision unable to accurately describe the classification ability of the network, so we do not use the $F1$-score, which contains the Precision. In addition, this paper will not add any fault-tolerant means.

4. Evaluation

4.1. Ablation Experiments. In this part, we mainly introduce the structure of the ARFNet and verify the effectiveness of the network through ablation experiments.

As mentioned in RepVGG [29], single channel architecture has the advantages of saving video memory and good flexibility. At the same time, the calculation density of 3×3 convolution on GPU is better than other parameters. Therefore, this paper selects VGG network model as the baseline (the specific structure is shown in Table 2). In addition, a multilayer perceptron (MLP) is also added to construct a small classifier to predict the existence distribution of four lanes (see Figure 7).

Figure 8 shows the ARF-module we used in our experiment. In the following article, in order to distinguish it from normal convolution, the receptive field adaptive module will be marked as A-conv, r is the dilation rate, and C is the number of feature maps.

To verify the effectiveness of the module, we add Se-block to convolution in the normal VGG network as the control group. The Tables 3 and 4is the comparison of the experimental result.

We fixed the number of steps to ensure that the training degree of the model is consistent. Through the experiment, we find that although the two models show high Accuracy, their renderings (see Figure 9) still fail to meet the expectations. This is because the visual judgment will be more affected if lane line pixels are misjudged as background. Based on this phenomenon, we give priority to the Recall and then consider the value of Accuracy.

It can be seen from the data in the Table and the renderings that the normal network like VGG-SENet presents a large number of misjudgments for lane pixels, which makes the lane line broken obviously. However, ARFNet uses its information learning ability of different receptive fields to improve this phenomenon.

In order to get better segmentation effect, we appropriately increase the number of feature maps in the module. Moreover, a weighted linear connector is added to enhance learning ability for aggregation. The modified ARF-module is shown as Figure 10 and the result of ablation experiment is shown in Tables 5 and 6.

In our evaluation method, the module with linear connector is much better than the module with only increasing feature maps. In addition, we compare $r = 1$ and $r = k$ (k is taken as 3 in this paper) to prove that the aggregation effect of different receptive fields is better than the aggregation of single receptive field. The result is shown in Tables 7 and 8. It is clear that, after different receptive field aggregation and adaptive process, the detection performance of each lane has been improved to a certain extent.

4.2. Adaptive Receptive Field Exploration Experiment. In this part, we make detailed experiments on the internal structure of ARF-module to find the best way to build it.

4.2.1. The Way of Feature Aggregation. In addition to the direct concatenation in the channel direction, different receptive field features can also be aggregation by summation. According to the description of the theory part (see Figure 3 and equation (2), the differences between the two are shown as

Table 2: ARFNet.

Layer	Type	Output-size
1	Conv	$216 \times 800 \times 64$
2	Maxpool	$108 \times 400 \times 64$
3	A-conv($r = 2$)	$108 \times 400 \times 64$
4	Conv	$108 \times 400 \times 128$
5	Maxpool	$54 \times 200 \times 128$
6	A-conv($r = 2$)	$54 \times 200 \times 128$
7	Conv	$54 \times 200 \times 256$
8	Maxpool	$27 \times 100 \times 256$
9	A-conv($r = 2$)	$27 \times 100 \times 256$
10	Conv	$27 \times 100 \times 512$
11	A-conv($r = 2$)	$27 \times 100 \times 512$
12	A-conv($r = 2$)	$27 \times 100 \times 512$
13	A-conv($r = 2$)	$27 \times 100 \times 512$
14	Conv	$27 \times 100 \times 128$
15	Conv	$27 \times 100 \times 5$
16	Upsample	$216 \times 800 \times 5$

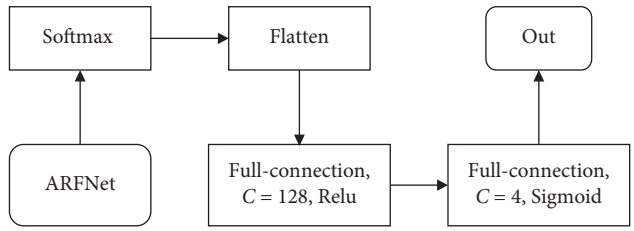


Figure 7: Lane existence prediction module.

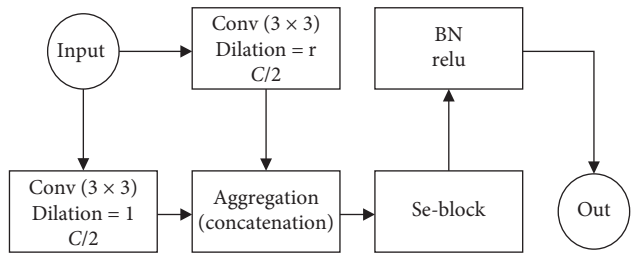


Figure 8: A-conv.

Table 3: Training results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
VGG-SE	0.4916	0.793	0.663	0.628	0.764	0.712	**0.918**	8000
ARFNet	**0.4817**	**0.808**	**0.679**	**0.639**	**0.775**	**0.725**	0.916	8000

Table 4: Test results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
VGG-SE	0.5007	**0.717**	0.614	0.582	0.703	0.654	**0.924**	900
ARFNet	**0.4534**	0.704	**0.700**	**0.688**	**0.762**	**0.713**	0.914	900

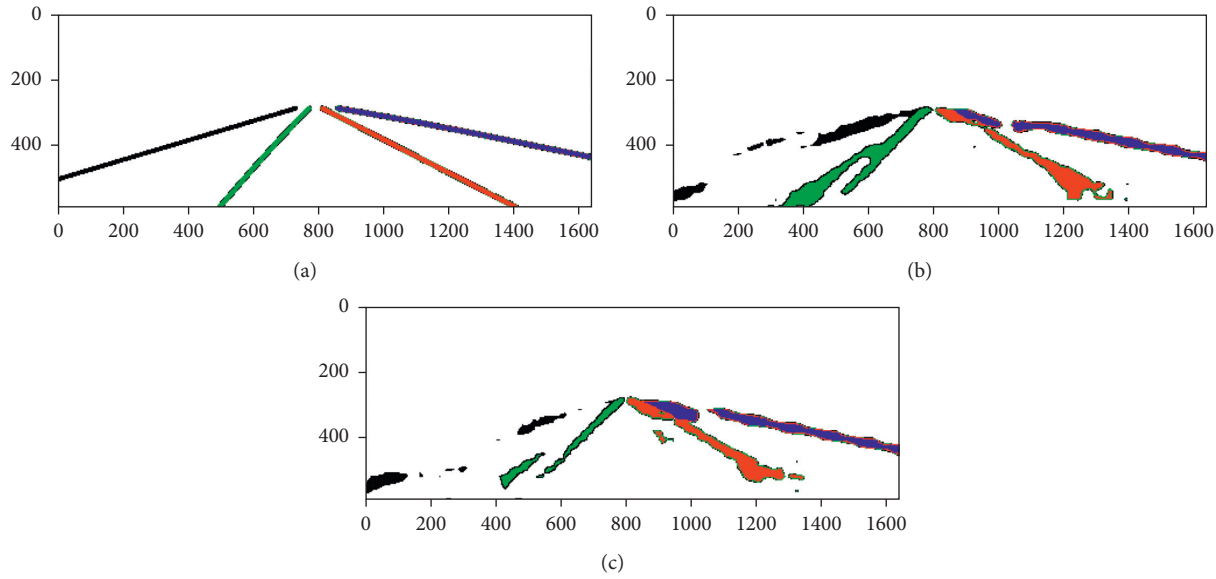


Figure 9: (a) Label. (b) ARFNet. (c) VGG-SENet.

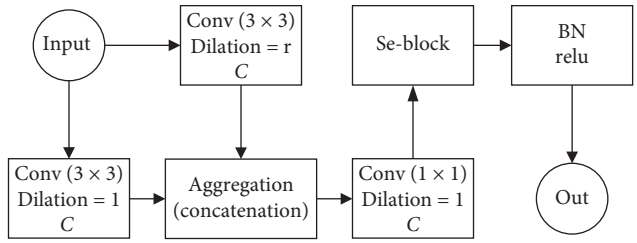


Figure 10: Modified ARF-module.

Table 5: Training results (the dilation rate is 3).

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
With linear connector	**0.4576**	**0.823**	**0.697**	**0.669**	**0.796**	**0.746**	0.916	8000
Without linear connector	0.4663	0.819	0.695	0.651	0.788	0.738	0.916	8000

Table 6: Test results (the dilation rate is 3).

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
With linear connector	**0.4275**	**0.812**	**0.755**	**0.709**	**0.833**	**0.777**	0.907	900
Without linear connector	0.4522	0.774	0.631	0.643	0.796	0.711	**0.922**	900

Table 7: Training results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
$r = 1$	0.4889	0.801	0.664	0.624	0.767	0.714	0.918	8000
$r = 3$	0.4576	0.823	0.697	0.669	0.796	0.746	0.916	8000

Table 8: Test results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
$r = 1$	0.4586	0.794	0.617	0.609	0.771	0.698	**0.914**	900
$r = 3$	**0.4275**	**0.812**	**0.755**	**0.709**	**0.833**	**0.777**	0.907	900

$$Y_{\text{Concatenation}} = \alpha\left(X_1 W_1 + X_2 W_2\right),$$
$$Y_{\text{Summation}} = \alpha\left(X_1 + X_2\right) W, \tag{12}$$

where Y is the output of the module, a is the score tensor provided by SE-block, X is the feature under different receptive fields, and W represents kernels in the linear connector. This method can greatly reduce the number of parameters and improve the running speed as well. The ARF-module with summation is shown in Figure 11 and the result is shown in Tables 9 and 10.

Obviously, this result is consistent with the results described in the paper of ResNet [30, 31] and DenseNet [32] that, at the cost of weakening the performance, the summation module improves the speed of the network and reduces the amount of calculation.

4.2.2. Optimization of Different Receptive Fields. Considering the factors of hardware and running speed, this paper will use the summation module in the optimal receptive field combination experiment. The receptive field combination in the summation module can be extended to

$$\alpha * x\left(r_1\right) + \beta * x\left(r_2\right) + \delta * i, \tag{13}$$

where α, β, δ represent the weight coefficients, r represents the dilation rate, i represents the input tensor, and x represents feature maps in the parallel structure. The experimental result is shown in Tables 11 and 12. In order to facilitate the analysis, we fix some parameters in the parallel structure, so that the first part does not occur dilation convolution, and we set all the coefficients to 1.

Through the experiment, we find that when the dilation rate exceeds 3, the Recall decreases obviously, but the Accuracy still increases. Therefore, we only use the result of $r = 4$ to represent $r > 3$ and do not show other results of $r > 4$. The renderings are shown in Figure 12.

It can be seen from the renderings that the resistance to interference information is only reflected in a certain appropriate range of dilation rate. That means despite the increase of r, the number of lane line pixels correctly predicted has increased significantly the number of background pixels misjudged as lanes has increased as well. Hence, we choose $r = 3$, which has relatively better evaluation result and rendering, for further experiments (see Figure 13).

Combined with the original data, we find that the double line of Lane 2 is not shown in the label (see Figure 9(a)). In this case, the segmentation renderings can directly reflect the quality of the results. From the segmentation renderings, the effect of $x\ (1) + x\ (3) - \mathrm{i}$ is much smoother and clearer. However, there are obvious lane disappearance and distortion of straight line model in the prediction of vehicle occlusion. Compared with $r = 3$ and $r = 4$, the ARF-model with $x\,(1) + x\,(3) - i$ has weaker generalization ability for lane lines when facing occlusion. Based on the above analysis, the combination of $r = 3$ and $r = 1$ is more suitable for CuLane dataset than other combinations.

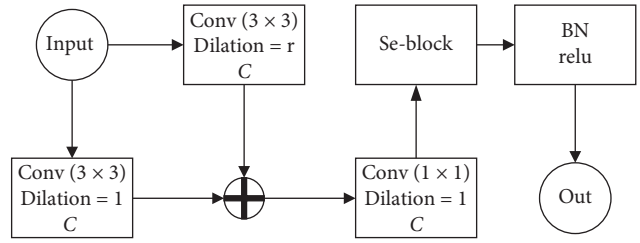


Figure 11: Summation module.

4.3. Results. In this part, the optimized ARFNet is compared with current SOTA models.

In the test, we choose the structure shown in Figure 10 and set the dilation rate to 3 as the final version of ARFNet. Since there is no detailed public data of Recall and Accuracy in CuLane dataset, we will use a unified training strategy to train each model. Considering the limitation of hardware, we only test some lightweight SOTA models: ResNet-34 [30, 31], SCNN [2]. The experimental result is shown in Table 13 and the comparison of renderings with SCNN is shown in Figure 14 (the first graph is the original data, the second graph is the segmentation result of ARFNet, and the third graph is the segmentation result of SCNN).

We selected four typical difficult scenarios for comparison. In result 1, the original data has four lane lines and are partially occluded by the tree shadow. In result 2, there are three clear lane lines, but the shooting angle is slightly different. In result 3, the original data has serious occlusion from people and vehicles. In result 4, the original data has obvious glass reflection, which causes a big trouble to predict the rightmost lane.

From the experimental results, ARFNet has achieved relatively good results but still cannot exceed SCNN to reach the level of SOTA now. According to the comparative analysis of the renderings, we have found problems that ARFNet shows typical attention preference. Though it can well judge the obvious lane line pixels, it is extremely sensitive to occlusions. The network tends to distinguish the covering part as background and only segments those pixels with distinct lane characteristics, which reflects that the robustness and the reasoning ability of the network for global context are still defective. We will continue to update the network in the following experiments to improve the current deficiencies.

Finally, we take 9×9 convolution kernel to check the segmentation map for smoothing filtering and add spline to select the appropriate prediction pixel points in order to bring better renderings (see Figure 15).

Table 9: Training results (dilation rate is 3).

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
Summation	0.4642	0.820	0.695	0.654	0.790	0.740	0.915	8000
Concatenation	**0.4576**	**0.823**	**0.697**	**0.669**	**0.796**	**0.746**	**0.916**	8000

Table 10: Test results (dilation rate is 3).

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
Summation	0.4367	0.791	0.697	0.668	0.820	0.744	0.909	900
Concatenation	0.4275	0.812	0.755	0.709	0.833	0.777	0.907	900

Table 11: Training results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
$x(1) + x(2)$	0.4742	0.815	0.685	0.643	0.780	0.731	**0.916**	8000
$x(1) + x(3)$	0.4642	0.820	0.695	0.654	0.790	0.740	0.915	8000
$x(1) + x(3)+i$	0.4869	0.803	0.678	0.632	0.768	0.721	0.915	8000
$x(1) + x(3)$-i	0.4740	0.818	0.683	0.652	0.783	0.734	0.914	8000
$x(1) + x(4)$	**0.4609**	**0.823**	**0.702**	**0.661**	**0.797**	**0.746**	0.913	8000

Table 12: Test results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
$x(1) + x(2)$	0.4623	0.757	0.673	0.628	**0.837**	0.724	0.915	900
$x(1) + x(3)$	**0.4367**	**0.791**	0.697	**0.668**	0.820	**0.744**	0.909	900
$x(1) + x(3)+i$	0.4621	0.765	**0.720**	0.570	0.755	0.702	0.917	900
$x(1) + x(3)$-i	0.4412	0.781	0.644	0.652	0.776	0.713	0.913	900
$x(1) + x(4)$	0.4436	0.742	0.658	0.611	0.810	0.705	**0.920**	900

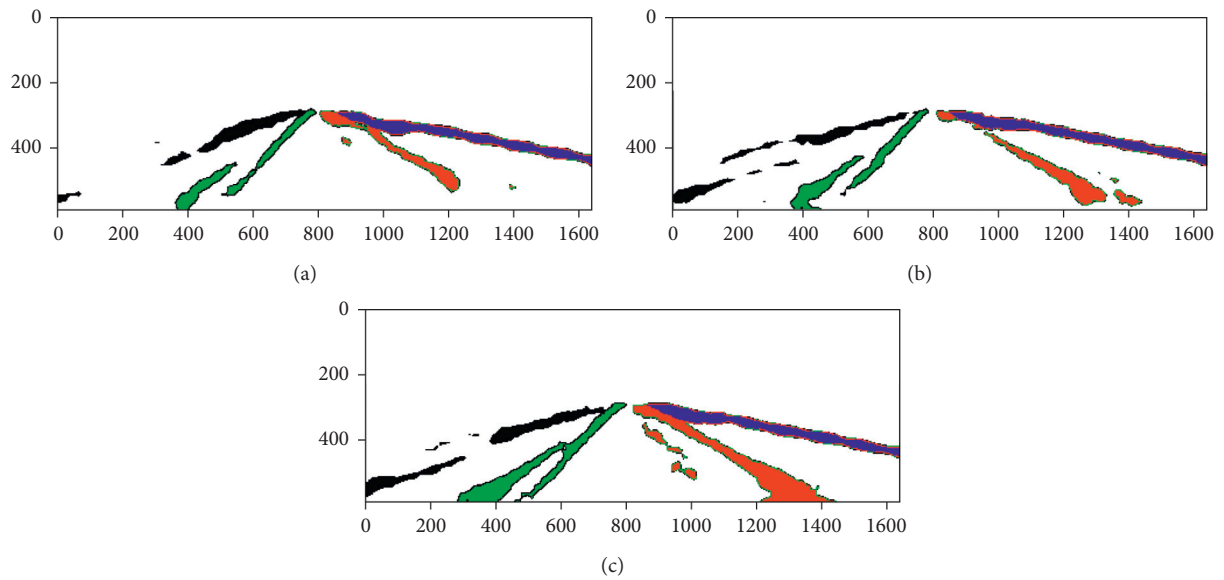


Figure 12: (a) The result of $r = 2$. (b) The result of $r = 3$. (c) The result of $r = 4$.

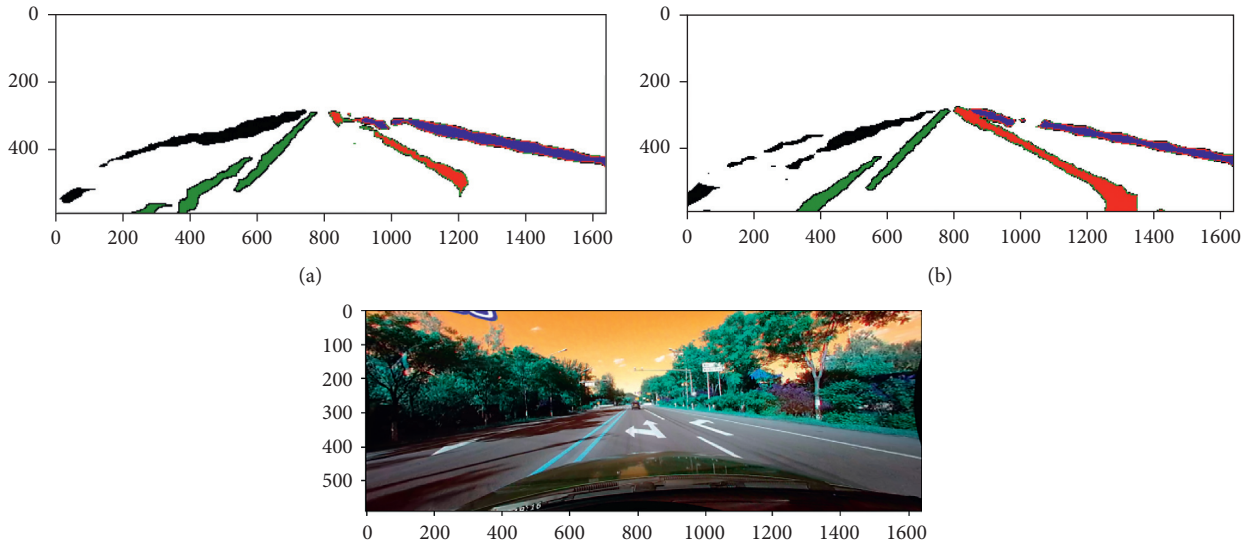


Figure 13: (a) The result of $x(1) + x(3) + i$. (b) The result of $x(1) + x(3) - i$. (c) Original data.

Table 13: Test results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
SCNN	**0.3391**	**0.881**	**0.820**	**0.797**	**0.863**	**0.840**	0.934	900
ARFNet	0.3725	0.843	0.780	0.753	0.824	0.800	**0.941**	900
ResNet-34	0.4150	0.834a	0.721	0.751	0.838	0.786	0.919	900

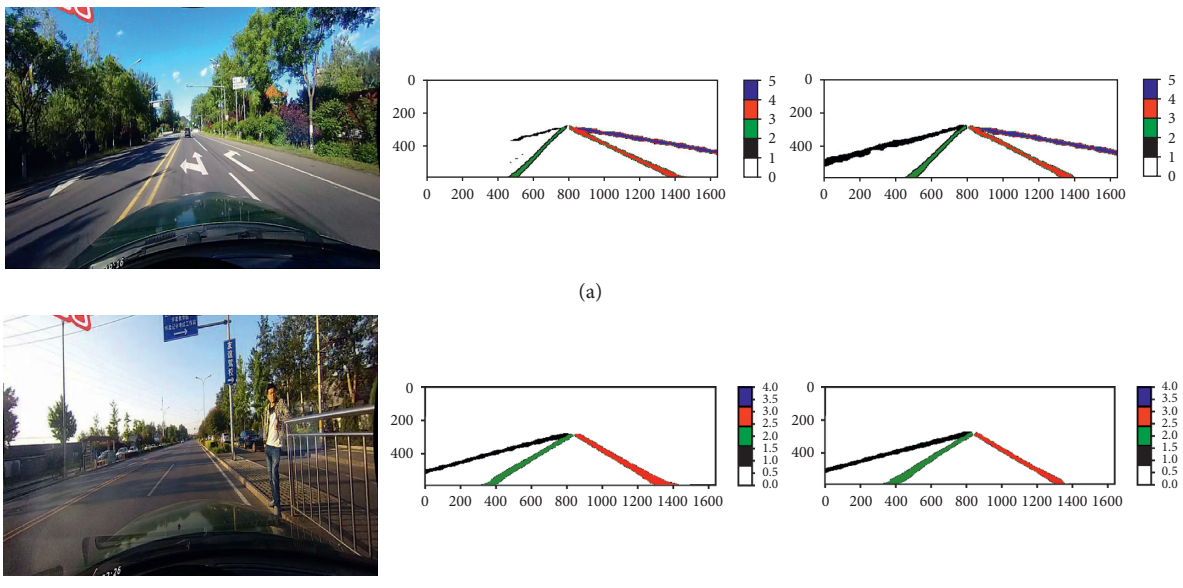
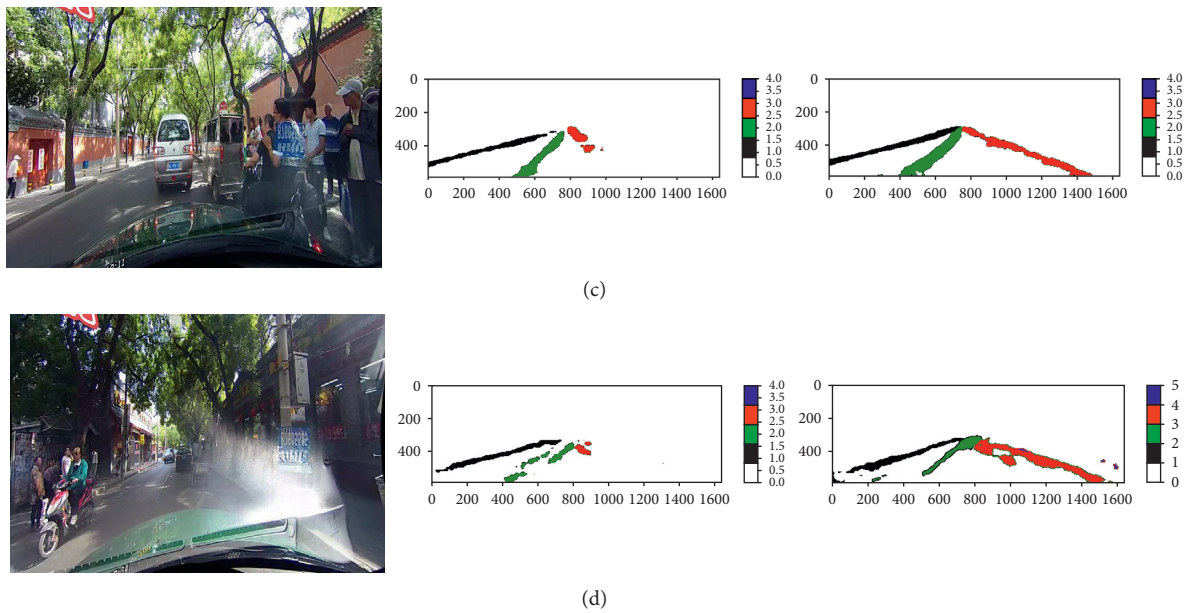


Figure 14: Continued.



(c)

(d)

Figure 14: (a) Result 1. (b) Result 2. (c) Result 3. (d) Result 4.

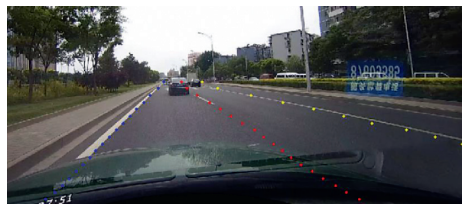


Figure 15: Lane prediction map postprocessing.

(a) (b)

Figure 16: (a) Method 1. (b) Method 2.

Table 14: Training results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
Method 1	0.4599	0.821	**0.698**	0.653	0.791	0.741	0.916	8000
Method 2	**0.4576**	**0.823**	0.697	**0.669**	**0.796**	**0.746**	0.916	8000

Table 15: Test results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
Method 1	0.4366	**0.817**	0.677	0.647	**0.855**	0.749	**0.915**	900
Method 2	**0.4276**	0.812	**0.755**	**0.709**	0.833	**0.777**	0.907	900

Table 16: Training results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3r(recall)	Lane 4 (recall)	Average	Accuracy	Step
Structure 1	0.4814	0.809	0.679	0.632	0.777	0.724	0.915	8000
Structure 2	0.4578	0.818	0.692	0.659	0.788	0.739	**0.918**	8000
Original structure	**0.4576**	**0.823**	**0.697**	**0.669**	**0.796**	**0.746**	0.916	8000

Table 17: Test results.

Name	Loss	Lane 1 (recall)	Lane 2 (recall)	Lane 3 (recall)	Lane 4 (recall)	Average	Accuracy	Step
Structure 1	0.4442	**0.821**	0.719	0.657	0.809	0.752	0.903	900
Structure 2	0.4331	0.753	0.669	0.667	0.796	0.721	**0.923**	900
Original structure	**0.4276**	0.812	**0.755**	**0.709**	**0.833**	**0.777**	0.907	900

Table 18: Structure 1.

Layer	Type	Output-size
1	Conv	$216 \times 800 \times 64$
2	Maxpool	$108 \times 400 \times 64$
3	A-conv($r = 3$)	$108 \times 400 \times 64$
4	Conv	$108 \times 400 \times 128$
5	Maxpool	$54 \times 200 \times 128$
6	A-conv($r = 3$)	$54 \times 200 \times 128$
7	Conv	$54 \times 200 \times 256$
8	Maxpool	$27 \times 100 \times 256$
9	A-conv($r = 3$)	$27 \times 100 \times 256$
10	Conv	$27 \times 100 \times 512$
11	Conv	$27 \times 100 \times 512$
12	Conv	$27 \times 100 \times 512$
13	Conv	$27 \times 100 \times 512$
14	Conv	$27 \times 100 \times 128$
15	Conv	$27 \times 100 \times 5$
16	Upsample	$216 \times 800 \times 5$

Table 19: Structure 2.

Layer	Type	Output-size
1	Conv	$216 \times 800 \times 64$
2	Maxpool	$108 \times 400 \times 64$
3	Conv	$108 \times 400 \times 64$
4	Conv	$108 \times 400 \times 128$
5	Maxpool	$54 \times 200 \times 128$
6	Conv	$54 \times 200 \times 128$
7	Conv	$54 \times 200 \times 256$
8	Maxpool	$27 \times 100 \times 256$
9	Conv	$27 \times 100 \times 256$
10	Conv	$27 \times 100 \times 512$
11	A-conv($r = 3$)	$27 \times 100 \times 512$
12	A-conv($r = 3$)	$27 \times 100 \times 512$
13	A-conv($r = 3$)	$27 \times 100 \times 512$
14	Conv	$27 \times 100 \times 128$
15	Conv	$27 \times 100 \times 5$
16	Upsample	$216 \times 800 \times 5$

5. Conclusion

In this study, we propose ARF-module to realize the aggregation and adaptation of multiple receptive fields. Although it has achieved relatively good results in CuLane dataset by picking up information from different receptive fields, it cannot reach the level of SOTA now. This is mainly reflected in the lack of prediction generalization ability when dealing with large area occlusion. In other words, the model relies too much on the lane line pixels in the image, which weakens the ability to extend the existing pixels to the line or curve model. We will further optimize ARFNet to solve this problem and try to realize ARFNet mentioned in theory.

Appendix

A. Module Optimization Experiment

When we add linear connectors to ARF-module, we consider that there are two structures (see Figure 16) to enhance the adaptive ability. The comparison result is shown in Tables 14 and 15.

Although the fitting degrees of the training set are similar, the network can better adapt to the information through the structure of Method 2. Therefore, we adopt this structure in this paper.

B. Structural Test

In previous experiments, receptive field aggregation occurred in each layer of the network. Therefore, it is unknown for its importance at different resolutions or layer with different width. In this case, the structural test is taken to optimize the overall structure of ARFNet. The specific structures are shown in Tables 16 and 17 and the experimental result is shown in Tables 18 and 19.

In structure 1, ARF-modules are only placed in high resolution and small width layers (bottom layers), while in structure 2, they are placed in low resolution and large width layers (top layers). Through experiments, we find that the original structure is still the best so that, in this paper, we will continue to use this structure. But interestingly, structure 1 and structure 2 show two different effects. From the difference between the results of Recall and Accuracy, modules working at bottom layers are more effective to improve the lane pixel prediction ability, while modules working at the top are more effective for background pixels.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The research work was supported by National Nature Science Foundation of China (No: 61705109), National Nature Science Foundation of China (No: 61931004), and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Jiangsu Province College Students Practice and Jiangsu Innovation & Entrepreneurship Group Talents Plan.

References

[1] S. Lee, J. Kim, J. S. Yoon et al., "Vpgnet: vanishing point guided network for lane and road marking detection and recognition,"vol. 1, pp. 1965–1973, in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, , IEEE, Venice, Italy, October 2017.

[2] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: spatial CNN for traffic scene understanding," *Association for the Advancement of Artifificial Intelligence*, vol. 1, no. 2, pp. 3–10, 2018.

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

[4] Y. Hou, Z. Ma, C. Liu, and C. Change Loy, "Learning lightweight lane detection CNNs by self attention distillation," 2019, http://arxiv.org/abs/1908.00821.

[5] M. Lee, J. Lee, D. Lee, W. Kim, S. Hwang, and S. Lee, "Robust lane detection via expanded self attention," 2021, http://arxiv.org/abs/2102.07037.

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

[7] N. Garnett, R. Cohen, T. Pe'er, R. Lahav, and D. Levi, "3D-LaneNet: end-to-end 3D multiple lane detection," 2019, http://arxiv.org/abs/1811.10203.

[8] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, "End-to-end lane shape prediction with transformers," 2020, http://arxiv.org/abs/2011.04233.

[9] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," 2017, http://arxiv.org/abs/1708.06519.

[10] M. Tan and Q. V. Le, "Efficientnet: rethinking model scaling for convolutional neural networks," 2019, http://arxiv.org/abs/1905.11946.

[11] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: scalable and efficient object detection," 2019, http://arxiv.org/abs/1911.09070.

[12] Y. Bengio and X. Glorot, "Understanding the difficulty of training deep feed forward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 249–256, DIRO, Universite de Montreal, Montreal, Quebec, Canada, 2010.

[13] Y. Zhang, J. Zhang, Q. Wang, and Z. Zhong, "DyNet: dynamic convolution for accelerating convolutional neural networks," 2020, http://arxiv.org/abs/2004.10694.

[14] J. Dai, H. Qi, Y. Xiong et al., "Deformable convolutional networks," 2017, http://arxiv.org/abs/1703.06211.

[15] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets V2: more deformable, better results," 2018, http://arxiv.org/abs/1811.11168.

[16] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: regularization strategy to train strong classifiers with localizable features," 2019, http://arxiv.org/abs/1905.04899.

[17] D. Ho, E. Liang, I. Stoica, P. Abbeel, and X. Chen, "Population based augmentation: efficient learning of augmentation policy schedules," 2019, http://arxiv.org/abs/1905.05393.

[18] E. D. Cubuk, B. Zoph, V. Vasudevan, Q. V. Le, and D. Mane, "Autoaugment: learning augmentation policies from data," 2018, http://arxiv.org/abs/1805.09501.

[19] Y. Wang, G. Huang, S. Song, X. Pan, Y. Xia, and C. Wu, "Regularizing deep networks with semantic data augmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016, http://arxiv.org/abs/1602.07261.

[21] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, http://arxiv.org/abs/1511.07122.

[22] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," 2019, http://arxiv.org/abs/1903.06586.

[23] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2017, http://arxiv.org/abs/1709.01507.

[24] J. Liu, C. Li, F. Liang et al., "Inception convolution with efficient dilation search," 2020, http://arxiv.org/abs/2012.13587.

[25] CULane [DB/OL], https://xingangpan.github.io/projects/CULane.html.

[26] B Carpenter, "Lazy sparse stochastic gradient descent for regularized multinomial logistic regression," Alias-i, LingPipe, Technical report, 2008.

[27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, http://arxiv.org/abs/1711.05101.

[28] I. Sutskever, J. Martens, G. Dahl, and G. Hinton in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013 (International Machine Learning Society (IMLS)*, pp. 2176–2184, 2013.

[29] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVGG: making VGG-style convNets great again," 2021, http://arxiv.org/abs/2101.03697.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, http://arxiv.org/abs/1512.03385.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the European Conference on Computer Vision*, Amsterdam, The Netherlands, September 2016.

[32] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017.