A Faster and Lightweight Lane Detection Method in Complex Scenarios

Shuaiqi Nie, Guiheng Zhang, Libo Yun and Shuxian Liu *

School of Computer Science and Technology, Xinjiang University, Ürümqi 830046, China; 107552204054@stu.xju.edu.cn (S.N.); 107552204118@stu.xju.edu.cn (G.Z.); ylb@stu.xju.edu.cn (L.Y.) * Correspondence: liushuxian@xju.edu.cn

Abstract: Lane detection is a crucial visual perception task in the field of autonomous driving, serving as one of the core modules in advanced driver assistance systems (ADASs). To address the insufficient real-time performance of current segmentation-based models and the conflict between the demand for high inference speed and the excessive parameters in resource-constrained edge devices (such as onboard hardware, mobile terminals, etc.) in complex real-world scenarios, this paper proposes an efficient and lightweight auxiliary branch network (CBGA-Auxiliary) to tackle these issues. Firstly, to enhance the model's capability to extract feature information in complex scenarios, a row anchor-based feature extraction method based on global features was adopted. Secondly, employing ResNet as the backbone network and CBGA (Conv-Bn-GELU-SE Attention) as the fundamental module, we formed the auxiliary segmentation network, significantly enhancing the segmentation training speed of the model. Additionally, we replaced the standard convolutions in the branch network with lightweight GhostConv convolutions. This reduced the parameters and computational complexity while maintaining accuracy. Finally, an additional enhanced structural loss function was introduced to compensate for the structural defect loss issue inherent in the row anchor-based method, further improving the detection accuracy. The model underwent extensive experimentation on the Tusimple dataset and the CULane dataset, which encompass various road scenarios. The experimental results indicate that the model achieved the highest F1 scores of 96.1% and 71.0% on the Tusimple and CULane datasets, respectively. At a resolution of 288 × 800, the ResNet18 and ResNet34 models achieved maximum inference speeds of 410FPS and 280FPS, respectively. Compared to existing SOTA models, it demonstrates a significant advantage in terms of inference speed. The model achieved a good balance between accuracy and inference speed, making it suitable for deployment on edge devices and validates the effectiveness of the model.

Keywords: lane detection; auxiliary attention; row anchor-based method; structure loss

1. Introduction

Lane detection plays a crucial role in the development of today's autonomous driving technology. Lane detection is increasingly being applied in the driving domain, driving the advancement of advanced driver assistance system (ADAS) technology. Lane detection, as one of the fundamental tasks in autonomous driving, plays a crucial role in applications such as adaptive cruise control, automatic steering, lane-keeping assistance, blind spot assistance, lane departure warning, and lane detection.

Currently, there are numerous efficient lane detection models and algorithms [1] available. However, lane detection models in the field of deep learning still face many challenges. As shown in Figure 1, in some complex scenarios (such as strong light exposure, poor night-time lighting, obstacles blocking the view, blurry lane lines, etc.), the problem of missing semantic information obtained by the model leads to poor detection.
performance. Furthermore, another challenge arises when applying lane detection models to resource-constrained edge devices, which imposes strict requirements for real-time performance and lightweight design. Many state-of-the-art (SOTA) models with high accuracy often have high parameter counts and inference times, making it difficult to simultaneously meet the real-time and lightweight requirements in practical scenarios. As a result, it is challenging to achieve an effective balance between accuracy and inference speed.

Figure 1. Several challenging lane detection scenarios (the different lane lines are annotated with different colors).

To address the current challenges, lane detection methods need to conduct higher-level semantic analyses of the lanes. Deep segmentation methods, with their stronger semantic information representation capabilities in image processing, are gradually becoming mainstream. For example, the SCNN [2] network achieves outstanding performance and robustness in semantic segmentation tasks across different scenarios through spatial perception, multi-scale processing, modular design, contextual awareness, and end-to-end training. However, due to the dense per-pixel transmission, this mechanism hampers the inference speed of the network, resulting in the need for higher computational costs and increased overhead. ENet-SAD [3] adopts a lightweight network architecture design. By allowing context information to be preserved in deeper network layers and enabling information propagation within these layers, it introduces convolution and pooling operations at different scales to achieve comprehensive lane line regression. This lightweight design enables ENet-SAD to operate efficiently on edge devices (embedded devices or mobile terminals). CLRNet [4] employs an FPN structure to improve its model’s ability to capture global features by pre-extracting lane features through the ROI module. CLRNet adopts a cross-fusion strategy, merging features from different levels through cross-level feature fusion to improve the accuracy and robustness of semantic segmentation. CondLaneNet [5] adopts FPN and self-attention mechanisms for feature extraction and introduces conditional convolution technology, which incorporates additional supplementary information into the convolution operation to enhance adaptability to different
scenarios. However, since the conditional convolution module is introduced to address issues in various scenarios, inaccurate or missing conditional information may impact the model’s detection performance.

Traditional segmentation-based lane detection models have traditionally prioritized accuracy. For example, Lu et al. [6] proposed a new semantic segmentation network capable of effectively segmenting keyframes to ensure the model captures many important semantic features in the image. While this method achieved good performance, it classifies segmentation pixel by pixel, resulting in limited real-time performance of the model. Recently, lane detection methods based on row anchors have become the preferred solution for addressing real-time constraints. Models such as UFLD [7], UFLDv2 [8], and CondLaneNet [5] have demonstrated excellent inference speeds, showcasing high real-time performance. The feature extraction method based on row anchors divides the input image into an initial grid row by row and then classifies each grid within each row individually. This approach balances inference speed while maintaining accuracy.

In response to the aforementioned issues, this paper aims to design a solution that balances accuracy and high real-time performance. The goal is to enhance model accuracy while achieving faster inference speeds, as well as reducing computational complexity and parameters. The main contributions of this paper are summarized as follows:

1. We propose an efficient and lightweight auxiliary segmentation network (CBGA-Auxiliary). This network shares model parameters and weights with the backbone network, collaborating with the backbone network to complete the segmentation task, thereby enhancing the overall generalization ability and inference speed of the model. The auxiliary segmentation network only functions during the training phase and is not used during inference, thus not affecting the final inference speed.

2. We utilized lightweight GhostConv convolutions to replace standard convolutions in the auxiliary branch, addressing the additional parameters and computational overhead introduced with the auxiliary segmentation network. The addition of an extra auxiliary branch increases the model’s size, parameters, and computational complexity. The experimental results indicate that the replaced GBGA-Auxiliary effectively reduces the model’s parameter count while improving both accuracy and inference speed compared with the initial model.

3. We introduce additional structural loss function. This paper adopts a feature extraction method based on row anchors, which has certain structural deficiencies. Considering that the lanes are narrow and continuous, an additional similarity loss between the lanes is added to the original loss function, further improving the model’s detection accuracy.

4. The extensive experiments show that our model achieves good performance on the Tusimple and CULane datasets. Additionally, it maintains high inference speed without sacrificing accuracy, striking a good balance between accuracy and inference speed. This validates the effectiveness of the model.

2. Related Work

This section provides a brief introduction to traditional detection methods and recent lane detection methods based on deep learning.

2.1. Traditional Methods

Traditional detection methods rely on handcrafted operators for feature extraction, followed by linear fitting using techniques such as Hough transform [9] and random sampling [10]. However, manual feature extraction cannot adequately address the diversity of lanes in different scenarios, leading to a lack of robustness in traditional methods when applied to real-world scenarios. Feature extraction is one of the critical factors affecting lane detection performance. Therefore, many traditional methods require the preprocessing of images to ensure the quality of feature extraction for lane detection tasks. For
example, Ghanem et al. [11] proposed a road lane detection method based on geometric modeling, which included image processing, feature extraction, and line fitting models. Firstly, in the image processing stage, a region of interest (ROI) was used to remove the irrelevant objects from the lane markings. In the feature extraction step, the Canny method was employed to extract the edge features from the image, which was robust to noise. Secondly, line segments were extracted using the Hough transformation. Subsequently, the input was filtered using a standard deviation (SD) filter.

2.2. Deep Learning Methods

With the development of deep learning, an increasing number of methods based on deep neural networks [12] have shown superior performance in the field of lane detection. For example, VPGNet [13] constructs a comprehensive probabilistic graphical model by incorporating multi-view perception and integrating geometric semantic information to model relationships between different elements, further enhancing the accuracy and robustness of model perception. In [14], the authors utilized long short-term memory (LSTM) networks to handle the elongated lane features. Fast-Draw [15] predicts the direction of each lane point and then sequentially draws the lane lines to complete the prediction.

Dewangan et al. [16] proposed an encoder–decoder network architecture for semantic segmentation, utilizing a hybrid model based on UNet and ResNet. Munir et al. [17] combined deep learning algorithms with attention mechanisms to detect road lanes and proposed a lane detection method based on dynamic visual sensors (LDNet). R. Zhang et al. [1] proposed an instance segmentation-based lane recognition method called RS-Lane to address detection in complex scenarios. Zhang et al. [18] introduced a real-time lane recognition system based on attention strategies. Currently, lane detection methods based on deep neural networks mainly fall into the following four categories: segmentation-based methods, anchor-based methods, parameter prediction-based methods, and key point-based methods.

2.2.1. Segmentation-Based Methods

Segmentation-based lane detection methods [3,19–21] are the most common approach, widely used due to their superior performance and robustness. Early methods [21,22] typically employed multi-class classification strategies for lane instance recognition. As mentioned in the previous section, this approach is inflexible and time-consuming. Segmentation-based methods typically predict instance masks. Some studies [7,23] have indicated that describing lane lines as masks is inefficient because the focus of instance segmentation is to accurately classify each pixel grid rather than obtaining specific linear shapes. Therefore, to overcome this issue, anchor-based detection methods have been proposed.

2.2.2. Anchor-Based Methods

Anchor-based detection methods [24,25] define a set of predefined anchor points or candidate boxes in the image. A classifier is then used to determine whether each anchor point contains the target object, and regression is applied to estimate their positions and bounding boxes. Predefined anchors can reduce the impact of no visual cue problems, which is very helpful for detection in some occluded scenes, thus enhancing instance recognition capabilities. LaneATT [25] designs elongated-shaped anchors and demonstrates superior performance on multiple datasets. Anchor-based detection methods can be further divided into row anchor-based detection methods and line anchor-based detection methods. Unlike traditional anchor-based detection methods, row anchor-based methods pay more attention to the row morphology and positional information of lane lines in the image to improve the accuracy and stability of lane detection. For example, UFLD [7] adopts a row anchor-based detection method and innovatively introduces a
classification approach, significantly reducing computational costs and achieving ultra-high inference speed. In 2022, Qin et al. [8] proposed an enhanced version of UFLD, called UFLDv2, which adopts a hybrid anchor approach replacing row anchors. This model dynamically analyzes lane slopes to select row anchors and line anchors, further improving detection accuracy. The main idea of line anchor-based detection methods is to predict the offset between line anchors and ground truth using prior information about lane lines, thus obtaining accurate lane predictions. However, its real-time performance is not considered ideal. CLRNet improves its model’s ability to capture global features by pre-extracting lane features using an ROI module. Additionally, it introduces the line IOU loss function to further improve the model’s detection accuracy.

2.2.3. Parametric Prediction Methods

Parameter prediction-based lane detection methods utilize parameterized models to detect lanes. By parameterizing the description of lane lines, outputting curves represented by curve equations, and using the model to predict lane parameters, lane detection is achieved. LSTR [26] was the first to introduce transformers into lane detection tasks and achieved high inference speed. PolyLaneNet [27] was the first to propose the use of deep networks for regressing lane curve equations. However, parameter prediction methods are sensitive to parameter prediction errors, such as errors in higher-order coefficients, which may lead to changes in lane shape. Although these methods have faster inference speeds, they may struggle to achieve higher performance in terms of accuracy.

2.2.4. Key Point-Based Methods

Key point-based detection methods frame the lane line detection task as a set of key point prediction tasks. They utilize specific algorithms or techniques to detect key points in images, extract and analyze features around these key points, determine whether they belong to lane lines, and finally generate the lane shape and lane position to complete detection. PINet [19] utilizes a stacked hourglass network to predict key point positions. Qu et al. [28] proposed FOLOLane, which generates per-pixel heatmaps with the same resolution as the input image to obtain points on the lane. This model emphasizes modeling local patterns and achieves global prediction through a bottom-up approach. Wang et al. [29] proposed GANet, which adopts two branches, fusion confidence feature and offset feature, to improve local accuracy. Each key point predicts the corresponding lane by adding the coordinate offset to the lane starting point offset in parallel.

3. Method

3.1. Model

This article aimed to design an efficient and lightweight network that can simultaneously meet the real-time requirements and detection accuracy for lane line detection. Inspired by the UFLD network architecture, this paper used the ResNet network as the backbone network and adopted a row-based feature extraction method for feature extraction. Based on this, a highly efficient and lightweight auxiliary segmentation network was designed. The overall framework of this network model consists of three modules: the feature extraction module, the auxiliary segmentation network, and the classification network, as shown in Figure 2.
Data preprocessing
X1 X2 X3 X4
Input
ResNet blocks
Backbone
CBGA module ×5
Auxiliary branch
Avg pooling
Reshape
Select by row
Total loss function
Classification
CBGA module ×4
CBGA module ×3
Concat
Conv2d
GBGA module ×4
3×288×800
FC
Classification loss
Auxiliary segmentation loss
Sim loss
Output

Figure 2. Diagram of the overall architecture of the network model. The model is mainly divided into three parts, as shown in the figure. Firstly, the residual network (ResNet) is used as the feature extraction network. Then, classification and auxiliary segmentation tasks are performed separately in the classification network and the auxiliary segmentation network. In the segmentation task, three auxiliary branch heads obtain feature information at different levels from different residual blocks of the ResNet network. The segmentation task is completed through a segmentation network composed of CBGA modules. Finally, the three auxiliary branches are merged into the main branch, and the final segmentation task is completed using the main branch head composed of lightweight modules called the CBGA.

3.1.1. CBGA Module

Currently, many models achieve high detection accuracy but often sacrifice real-time performance and inference speed to improve accuracy. This is not friendly for deployment on edge devices. Therefore, this paper considers designing an efficient and lightweight auxiliary segmentation network to address this issue. For most models, injecting attention mechanisms into classification tasks can significantly improve the classification performance. Therefore, this paper considers designing a convolutional module with an attention mechanism called Conv-BN-GELU-SE, abbreviated as CBGA (as shown in Figure 3). The aim is to enhance the network’s focus on the details of the lane lines across various channel feature maps, thereby training a superior network model. The newly designed convolutional module in this paper fully leverages the advantages of the SE (squeeze and excitation) attention module. After the backbone network conducts standard convolutional operations on the feature maps, the SE attention module explicitly models the interdependencies between the convolutional feature channels to enhance the network’s representation capability. Especially for network models with ResNet as the backbone network, the proposed CBGA module is more effective. This is because ResNet is a residual network with deeper network layers. As the network depth increases, the convolutional operations become more complex, and feature extraction becomes more challenging. The CBGA module designed in this paper alleviates the problem of increased channel dependencies due to the increase in network depth by passing the feature information into the SE module after the convolution and normalization operations. In addition, weighting the convolutional weight matrix with the feature maps from the SE module further enhances the representational capacity of individual channel features. It also reduces inter-channel dependency issues while dynamically acquiring richer contextual feature information. Through this mechanism, the model can dynamically select feature information while learning global information and adaptively allocate weights for different channels, thereby effectively enhancing the network’s representation capability. Moreover, it
greatly improves the segmentation speed of the auxiliary segmentation network and enhances accuracy.

![Diagram](image)

**Figure 3.** Module design. (1) CBGA module. (2) GBGA module.

### 3.1.2. GhostConv Lightweight Module

We introduced the designed auxiliary segmentation network into the model, which greatly improved the training speed. However, it also increased the computational complexity of the model and the number of parameters in the model. Based on this, we considered replacing some of the lightweight convolutions in the auxiliary segmentation network to reduce the model’s complexity and the overall number of parameters. The Ghost module [30] is a lightweight design specifically aimed at convolutional neural networks and is suitable for embedded devices. The core idea of the Ghost module is to generate additional “ghost” feature maps using low-cost linear transformations based on existing feature maps. This approach improves the computational efficiency of the network while keeping the parameter count of the Ghost module low. The Ghost module is designed to enhance the efficiency of the network while maintaining a low parameter count, making it suitable for resource-constrained environments such as embedded devices. GhostConv mainly consists of three parts: regular convolution, Ghost generation, and the concatenation of feature maps (see Figure 4). Part A of the figure represents the regular convolution operation, while part B represents the GhostConv operation. The specific process is as follows: first, regular convolution is applied to obtain a series of feature maps, then a series of simple linear operations $\Phi_k \ (k = 1, 2, 3 \ldots)$ are applied to these feature maps to obtain a series of “ghost” feature maps. Finally, these feature maps are fused with the feature maps obtained from the first step of regular convolution through concatenation to obtain the final feature map output. Therefore, compared to relying on additional convolutional layers, GhostConv significantly reduces the required computational overhead and maximizes the utilization of the available computing and memory resources. To ensure the detection performance of the model, this paper only considered replacing the convolutions in the main branch. The replaced module is abbreviated as GBGA (GhostConv-BN-GELU-SE), as shown in Figure 3(2). Finally, this paper considered employing a hybrid network
consisting of the CBGA and GBGA modules for segmentation training. This approach effectively maintains a good balance between accuracy and real-time performance.

![Diagram](image)

**Figure 4.** Ghost module.

### 3.1.3. Auxiliary Attention Segmentation Branches

Based on the introduction of the above sections, the architecture of the auxiliary segmentation network designed in this paper consists of two parts: the auxiliary branch head and the main branch head (see Figure 5). Considering the deep hierarchical network structure of the residual network ResNet, we overlayed different numbers of CBGA convolution modules on the auxiliary branch head at different levels. Based on the introduction of the above sections, the structure of the auxiliary segmentation network designed in this paper consists of two parts: the auxiliary branch head and the main branch head (see Figure 5). Considering the deep hierarchical network structure of the residual network ResNet, we overlayed different numbers of CBGA convolution modules on the auxiliary branch head at different levels.

For the main branch head, this part needs to integrate the feature information provided by the preceding auxiliary branch heads and concatenate them into a complete feature map. Therefore, the acquired feature information is relatively extensive, leading to increased computational complexity. Consequently, in this section, employing lightweight CBGA modules on the main branch significantly reduces the model’s computational load. Additionally, replacing the standard convolutions with GhostConv in the main branch ensures real-time performance because it has fewer parameters, making the model more lightweight. In addition, in the main branch, using GhostConv convolution operations with dilation = n (n = 2, 4) helps integrate feature information in the feature maps with fewer computational resources. Simultaneously, it effectively increases the receptive field, further enhancing the model’s inference speed. This novel approach of the auxiliary segmentation network demonstrates significant performance in practical applications. Its superior inference speed meets the demanding requirements of real-time applications, injecting new vitality into the technological advancement of this field.
Figure 5. The structure of the auxiliary segmentation network figure. \(\times\) represents the stacked convolution and \(\oplus\) represents concatenation. The three auxiliary branch heads complete the feature extraction for different levels of residual blocks, and finally, the main branch combines them via concatenation. Lastly, the feature maps are upsampled to complete the segmentation training task (this auxiliary segmentation network only operates during the training phase and does not affect the final inference speed).

3.2. Loss Function

In this chapter, our model loss function can mainly be divided into four modules: auxiliary segmentation loss, classification loss, expectation loss, and the introduced similarity loss function.

For the expected loss \(L_{\text{exp}}\) and classification loss \(L_{\text{cls}}\), we assume that the \(h\) rows are predefined on the image, and the \(w\) network units are partitioned on each row (see Figure 6). The number of lane lines we need for detection is denoted as \(C\). \(X\) represents the global feature map of the input image, where \(f^{i,j}\) represents the position grid unit of the \(i\)-th lane line in the \(j\)-th row. Then, the probability prediction of the position relationship of the lane lines can be represented as Equation (1):

\[
p_{i,j} = f^{i,j}(X), i \in [1, C], j \in [1, h]
\] (1)

Figure 6. The relationship between the positions of lane points.

After obtaining the probability of the lane line positions, we considered using the predicted expected value as the approximate value of the lane grid points. We achieved this by applying the Softmax function to obtain the probability of each grid being a lane
line in each row. Then, we calculated the expectation of the probability of the grid units in each row and, finally, computed the coordinates of the lane points as shown in Equations (2) and (3). The final expected loss is represented as Equation (4):

\[ \text{Prob}_{i,j} = \text{soft max}(P_{i,j,k}) \]  

(2)

\[ \text{Loc}_{i,j} = \sum_{k=1}^{w} k \cdot \text{Prob}_{i,j,k} \]  

(3)

\[ L_{\text{exp}} = \sum_{i=1}^{C} \sum_{j=1}^{h} L_{i} \left( \text{Loc}_{i,j}, T_{i,j} \right) \]  

(4)

We divided the lane detection task into a row classification problem. Therefore, we need to predict the probability of each row containing a lane line without the need to individually classify the pixels. We can consider the classification loss as the sum of the cross-entropy between the predicted positions and the ground truth positions of each lane line in the predefined rows. Thus, the formula for the classification loss of the lane lines is Equation (5):

\[ L_{\text{cls}} = \sum_{j=1}^{h} \sum_{i=1}^{c} T_{i,j} \log(P_{i,j}) + (1 - T_{i,j}) \log(1 - P_{i,j}) \]  

(5)

In the equation, \( P_{i,j} \) represents the predicted probability of the lane line positions (as explained in Equation (1)), and \( T_{i,j} \) represents the label of the \( i \)-th lane line in the \( j \)-th row. If a lane line exists, \( T_{i,j} \) is 1; otherwise, it is 0.

For the additional structural loss function \( L_{\text{sim}} \) (similarity loss), we considered that the lane lines possess continuity characteristics, meaning that the lane points in adjacent row anchors should be close to each other; our method based on row anchors may have had some structural flaws in the position relationship loss between adjacent anchor points. These flaws could have impacted the detection performance of the model. Therefore, we introduced a similarity loss function to compensate for this deficiency. Specifically, as lane point position relationships are represented by classification vectors, we constrained the distribution density of the classification vectors on adjacent rows to achieve lane line continuity. In this way, the similarity loss function can be represented as Equation (5):

\[ L_{\text{sim}} = \sum_{i=1}^{c} \sum_{j=1}^{h-1} \left\| P_{i,j} - P_{i,j+1} \right\|_1 \]  

(6)

where \( P_{i,j} \) represents the prediction of the lane line in the \( j \)-th row. \( \left\| \cdot \right\|_1 \) denotes the L1 norm. For the auxiliary branch, cross-entropy is also used as the auxiliary segmentation loss \( L_{\text{seg}} \). The final loss function is the weighted average of the expected loss, classification loss, segmentation loss, and similarity loss function. The formula for the total loss calculation is shown in Equation (7), where \( \alpha, \beta, \gamma, \eta \) are the loss coefficients:

\[ L_{\text{total}} = \alpha L_{\text{cls}} + \beta L_{\text{seg}} + \gamma L_{\text{sim}} + \eta L_{\text{exp}} \]  

(7)

4. Experiment

4.1. Datasets

In our experimental evaluation, we conducted experiments on two publicly available benchmark datasets: Tusimple [31] and CULane [32]. These two datasets are commonly regarded as the main datasets in the region of lane line detection. Tusimple includes a
dataset of highway driving scenes with 3268 images in the training set, 358 images in the validation set, and 282 images in the test set. These images are standardized to 1280 × 720 pixels. CULane contains 133,235 images, with 88,880 images forming the training set, 34,680 images forming the test set, and the remaining 9675 images forming the validation set. The CULane dataset comprises data from nine different challenging scenarios, including congestion, night-time, no lane markings, shadows, intersections, and other real-world scenarios. The image resolution is standardized to 1640 × 590 pixels. Specific details of the two datasets are provided in Table 1.

Table 1. Dataset introduction. The “Fork” indicates whether dataset have a crossroads line.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Val</th>
<th>Test</th>
<th>Road Type</th>
<th>Fork</th>
<th>Scenarios</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tusimple</td>
<td>3.3 K</td>
<td>0.4 K</td>
<td>2.8 K</td>
<td>Highway</td>
<td>×</td>
<td>1</td>
<td>1280 × 720</td>
</tr>
<tr>
<td>CULane</td>
<td>88.9 K</td>
<td>9.7 K</td>
<td>34.7 K</td>
<td>Urban and highway</td>
<td>√</td>
<td>9</td>
<td>1640 × 590</td>
</tr>
</tbody>
</table>

4.2. Experimental Parameters and Environment

The model training for this paper was conducted on a Win11 operating system, utilizing an NVIDIA 3080 graphics card with 12GB of memory. Python was the programming language used, and the model was built under the PyTorch 1.12.1 framework. ResNet-18 and ResNet-34 were chosen as the backbone networks. After completing data preprocessing, the images were resized to 288 × 800 to fit the model input. In the training experiments on both datasets, we set our train batch size to 32. This size was chosen to sufficiently consider memory capacity while effectively utilizing computational resources, serving as the optimal threshold. With this batch size, the model exhibited fast convergence and high accuracy. Through experimentation, we ultimately set the initial learning rate to 4 × 10⁻⁴, the momentum to 0.9, and the weight decay to 1 × 10⁻⁴ (which is currently the most common practice). These settings aid in faster convergence during training and improve the model’s generalization ability. In the period of the training process, we utilized the Adam optimizer as the model optimization algorithm and employed cosine annealing learning rate scheduling to train the model. Through experimentation, we verified that when the number of training epochs reached 150 for Tusimple and 100 for CULane, the model’s accuracy stabilized. Therefore, according to our experimental results, we decided that setting the training epochs to 150 for Tusimple and 100 for CULane was reasonable. Specific results are presented in Figure 7.

Figure 7. Best training epochs on the two datasets.
4.3. Evaluation Metrics

Due to the differences in the datasets, the selected evaluation metrics varied between the two datasets. For the Tusimple dataset, the primary evaluation metric was accuracy. The formula for accuracy calculation is as follows:

$$\text{accuracy} = \frac{\sum C_{clip}}{\sum S_{clip}}$$ (8)

where $C_{clip}$ represents the number of correctly predicted lane points out of all the lane points, and $S_{clip}$ represents the total number of labeled lane points. For the CULane dataset, we selected the F1 measure as the evaluation metric. We computed the intersection over union (IoU) between the predicted values of the lane lines and the ground truth labels to determine the true positives (TPs), false positives (FPs), or false negatives (FNs) for each sample. We considered samples with an IoU threshold greater than 0.5 as true positive samples. The specific formulas are as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$ (9)

$$\text{Recall} = \frac{TP}{TP + FN}$$ (10)

$$F1\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$ (11)

4.4. Results

We evaluated ResNet-18 and ResNet-34 as the backbone networks on the Tusimple and CULane datasets. Additionally, in order to validate the effectiveness of our design model, we conducted comparative analyses with existing methods, including those based on row anchors, key points, SCNN, LaneNet, UFLD, and others. The results are shown in Tables 2 and 3. The experimental results indicate that the proposed method maintains high accuracy in various complex challenging scenarios. Additionally, it has fewer parameters, a faster inference speed, and lower computational complexity compared to existing state-of-the-art (SOTA) models. It exhibits significant advantages in terms of detection speed. To further intuitively demonstrate the detection performance of our model, we provided visualizations of the detection results. The visualization results for the Tusimple and CULane datasets are shown in Figure 8.

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Acc%</th>
<th>FP%</th>
<th>FN%</th>
<th>FPS</th>
<th>Params(M)</th>
<th>FLOPs(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNN [25]</td>
<td>95.97</td>
<td>96.53</td>
<td>6.17</td>
<td>1.80</td>
<td>7.5</td>
<td>20.27</td>
<td>328.4</td>
</tr>
<tr>
<td>LaneNet [20]</td>
<td>96.10</td>
<td>96.4</td>
<td>7.80</td>
<td>2.40</td>
<td>52</td>
<td>31.66</td>
<td>12475.8</td>
</tr>
<tr>
<td>LaneATT [29]</td>
<td>96.06</td>
<td>96.10</td>
<td>5.64</td>
<td>2.17</td>
<td>26</td>
<td>-</td>
<td>70.5</td>
</tr>
<tr>
<td>UFLD [7]</td>
<td>87.87</td>
<td>95.82</td>
<td>19.05</td>
<td>3.92</td>
<td>312</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CondLaneNet [5]</td>
<td>97.24</td>
<td>96.54</td>
<td>2.01</td>
<td>3.50</td>
<td>58</td>
<td>-</td>
<td>44.8</td>
</tr>
<tr>
<td>ResNet-18 [33]</td>
<td>92.34</td>
<td>92.69</td>
<td>9.48</td>
<td>8.22</td>
<td>-</td>
<td>11.69</td>
<td>0.91</td>
</tr>
<tr>
<td>ResNet-34 [33]</td>
<td>92.50</td>
<td>92.84</td>
<td>9.18</td>
<td>7.96</td>
<td>-</td>
<td>21.80</td>
<td>1.84</td>
</tr>
<tr>
<td>LNet [34]</td>
<td>94.38</td>
<td>94.43</td>
<td>11.5</td>
<td>5.3</td>
<td>143</td>
<td>2.07</td>
<td>228.2</td>
</tr>
<tr>
<td>ERFNet [35]</td>
<td>94.78</td>
<td>95.20</td>
<td>11.9</td>
<td>6.2</td>
<td>59</td>
<td>2.68</td>
<td>21.5</td>
</tr>
<tr>
<td>Ours (ResNet-18)</td>
<td>95.10</td>
<td>95.82</td>
<td>19.1</td>
<td>4.01</td>
<td>410</td>
<td>14.88</td>
<td>1.8</td>
</tr>
</tbody>
</table>
Table 3. Experimental comparison of the CULane dataset. For crossroad scenes, we only provide false positive samples. A lower number of false positive samples indicates better performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Total</th>
<th>Normal</th>
<th>Crowded</th>
<th>Dazzle</th>
<th>Shadow</th>
<th>No line</th>
<th>Curve</th>
<th>Crossroad</th>
<th>Arrow</th>
<th>Night</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNN [25]</td>
<td>71.6</td>
<td>96.0</td>
<td>58.5</td>
<td>58.5</td>
<td>43.4</td>
<td>69.7</td>
<td>64.4</td>
<td>1990</td>
<td>84.1</td>
<td>66.1</td>
</tr>
<tr>
<td>ERFNet [35]</td>
<td>73.1</td>
<td>91.5</td>
<td>71.6</td>
<td>66.0</td>
<td>71.3</td>
<td>45.1</td>
<td>66.3</td>
<td>2199</td>
<td>87.2</td>
<td>67.1</td>
</tr>
<tr>
<td>UFLD-Res18 [7]</td>
<td>68.4</td>
<td>87.7</td>
<td>66.0</td>
<td>58.4</td>
<td>63.8</td>
<td>40.2</td>
<td>57.9</td>
<td>1743</td>
<td>81.0</td>
<td>62.1</td>
</tr>
<tr>
<td>UFLD-Res34 [7]</td>
<td>72.3</td>
<td>90.7</td>
<td>70.2</td>
<td>59.5</td>
<td>69.3</td>
<td>44.4</td>
<td>69.5</td>
<td>2037</td>
<td>85.7</td>
<td>66.7</td>
</tr>
<tr>
<td>LNet [34]</td>
<td>74.1</td>
<td>92.7</td>
<td>72.9</td>
<td>61.9</td>
<td>65.7</td>
<td>40.1</td>
<td>57.8</td>
<td>2118</td>
<td>81.3</td>
<td>65.1</td>
</tr>
<tr>
<td>ResNet18-SAD [3]</td>
<td>70.5</td>
<td>89.8</td>
<td>68.1</td>
<td>59.8</td>
<td>67.5</td>
<td>42.5</td>
<td>65.5</td>
<td>1995</td>
<td>83.9</td>
<td>64.2</td>
</tr>
<tr>
<td>ResNet34-SAD [3]</td>
<td>70.7</td>
<td>89.9</td>
<td>68.1</td>
<td>59.9</td>
<td>67.7</td>
<td>42.2</td>
<td>66.0</td>
<td>1960</td>
<td>83.8</td>
<td>64.6</td>
</tr>
<tr>
<td>Ours (ResNet-18)</td>
<td>70.1</td>
<td>89.0</td>
<td>67.8</td>
<td>58.1</td>
<td>61.6</td>
<td>41.0</td>
<td>58.2</td>
<td>1741</td>
<td>84.0</td>
<td>63.7</td>
</tr>
<tr>
<td>Ours (ResNet-34)</td>
<td>71.0</td>
<td>90.8</td>
<td>70.8</td>
<td>61.6</td>
<td>71.4</td>
<td>44.5</td>
<td>65.1</td>
<td>2028</td>
<td>86.3</td>
<td>66.1</td>
</tr>
</tbody>
</table>

Our experimental comparison results of the TuSimple dataset are shown in Table 2. Our model’s inference speed was significantly higher than that of the compared models, with the fastest inference speed reaching up to 410 FPS, which was 98 FPS higher than UFLD. Additionally, our ResNet-34 version model achieved an F1 score of 96.10 at 277 FPS, with 23.39M parameters and 2.78G computational load. Compared to the baseline models, ResNet-18 and ResNet-34, both versions of our method achieved better detection results with lower parameter counts and computational complexity, making them more lightweight. Our ResNet-34 version, compared to the SCNN method with almost the same parameter count, achieved comparable detection performance but with a significantly higher inference speed of 277 FPS. Additionally, our model had low computational complexity, effectively alleviating the hardware load. Therefore, it exhibits significant advantages for deployment on some edge devices (such as embedded devices, mobile devices, etc.).

Our experimental results of the CULane dataset are shown in Tables 3 and 4. From both tables, it can be observed that our second version model achieved an inference speed of up to 334 FPS, surpassing the baseline ResNet-34 model by 3.6% in the F1 score, even with an additional 2.07 M parameters and 0.94 G computational complexity. Moreover, it achieved the best detection results in complex scenarios such as shadows and crossroads. Both versions of our method demonstrated significant improvements in real-time performance, with the lowest real-time performance reaching 280 FPS even in complex scenes of CULane. In summary, compared to mainstream models, our method maintains competitiveness in accuracy while exhibiting significant advantages in model detection speed and computational complexity. This demonstrates that our method can effectively balance accuracy and detection speed, and its lower computational complexity makes it suitable for deployment on some embedded devices.

Table 4. FPS, parameters, and FLOPs indicator comparative study in CULane.

<table>
<thead>
<tr>
<th>Method</th>
<th>FPS</th>
<th>Params (M)</th>
<th>FLOPs (G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNN [25]</td>
<td>7.5</td>
<td>20.27</td>
<td>328.40</td>
</tr>
<tr>
<td>ERFNet [35]</td>
<td>59</td>
<td>2.68</td>
<td>21.50</td>
</tr>
<tr>
<td>LNet [34]</td>
<td>143</td>
<td>2.07</td>
<td>22.49</td>
</tr>
<tr>
<td>CondLaneNet [5]</td>
<td>152</td>
<td>-</td>
<td>19.60</td>
</tr>
<tr>
<td>TSA-LNet [3]</td>
<td>142</td>
<td>2.28</td>
<td>47.02</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>-</td>
<td>21.80</td>
<td>1.84</td>
</tr>
<tr>
<td>Ours (ResNet-18)</td>
<td>334</td>
<td>14.88</td>
<td>1.80</td>
</tr>
<tr>
<td>Ours (ResNet-34)</td>
<td>280</td>
<td>23.39</td>
<td>2.78</td>
</tr>
</tbody>
</table>
As shown in Figure 8, our model demonstrates significant advantages in various complex scenarios. Firstly, the first row shows the detection results of the Tusimple dataset. It is evident that our model can accurately perform lane recognition even under vehicle occlusion and in complex curves. Additionally, with the ResNet-34 version, the inference speed can reach 277 FPS. Secondly, the second and third rows display our model’s detection performance across different scenarios on the CULane dataset. From the images, it is clear that our model still accurately completes detection, even in more complex scenarios. For instance, the second row lists the detection results under strong illumination, obstacle occlusion, and shadowed conditions. The third row presents the detection outcomes in night-time scenarios, including vehicle occlusion and situations where strong light causes the road to be blurry or even invisible. While maintaining accuracy, our ResNet-34 version can achieve an inference speed of 280 FPS. In conclusion, the analysis of the experimental results verifies the accuracy of our model’s detection performance in different scenarios and demonstrates that our model achieves a good balance between accuracy and inference speed.

**Figure 8.** Visualization results for both datasets. The first row displays the visualization results from the Tusimple dataset, while the subsequent two rows display the visualization results from the CULane dataset, including the detection results under various challenging scenarios, such as strong light, night, shadows, and occlusions.

### 4.5. Ablation Experiments

Through controlled experiments (see Table 5), we conducted ablation experiments on the auxiliary branch, Ghost module, and loss functions to validate the effectiveness of our model. From the results of the ablation experiments, we observed that when we adopted the CBGA-Auxiliary module, the inference speed of our model increased by 88 FPS, and the accuracy improved by 2.98%. However, the parameter count of our model increased to 25.75 M, and the computational complexity rose to 2.84 G. Therefore, by introducing the Ghost lightweight module, the experimental results showed that after replacing the lightweight module, the inference speed of our model increased by an additional 30 FPS, the parameter count decreased by 2.36 M, and the computational complexity decreased
by 0.8 G. Additionally, we observed a slight decrease in the model’s accuracy to 95.20%. Considering all factors, the Ghost module significantly improved the model’s inference speed and made it more lightweight with almost no sacrifice in accuracy. Finally, by introducing the enhanced structural loss function, the model achieved an accuracy of 96.10%. This section thoroughly verifies the effectiveness and rationality of the proposed model architecture in practical lane detection tasks.

Table 5. Ablation experiments conducted on the Tusimple dataset.

<table>
<thead>
<tr>
<th>Baseline</th>
<th>CBGA-Auxiliary</th>
<th>Ghost Module</th>
<th>New Loss</th>
<th>FPS</th>
<th>Params (M)</th>
<th>FLOPs (G)</th>
<th>Acc%</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>312</td>
<td>21.80</td>
<td>1.84</td>
<td>92.84</td>
<td>5.9</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>400</td>
<td>25.75</td>
<td>2.84</td>
<td>95.82</td>
<td>3.5</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>430</td>
<td>23.39</td>
<td>2.78</td>
<td>95.20</td>
<td>2.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>394</td>
<td>25.75</td>
<td>2.98</td>
<td>96.30</td>
<td>3.7</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>410</td>
<td>23.39</td>
<td>2.84</td>
<td>96.10</td>
<td>2.3</td>
</tr>
</tbody>
</table>

5. Conclusions

This paper proposes an efficient and lightweight lane detection model aimed at improving lane detection accuracy while achieving faster detection speed with fewer parameters and less computational complexity. To achieve this goal, a new auxiliary segmentation network is designed, and an innovative convolution module called CBGA is introduced as the basic module of the auxiliary segmentation network to enhance the model’s training speed. Secondly, to ensure the model is more lightweight, the Ghost lightweight module is adopted to replace some convolutions in the auxiliary segmentation network, addressing the issue of model parameterization. Additionally, using GhostConv convolution operations significantly reduces the model’s computational costs while almost not affecting the model’s detection accuracy. Finally, an additional similarity loss function is introduced to compensate for the structural defects of the row anchor method, further improving the model’s detection accuracy.

Finally, through experiments, we demonstrate that our model achieves outstanding performance on the Tusimple and CULane datasets while ensuring accuracy; the inference speed reaches up to 410 FPS and 334 FPS, respectively. Additionally, compared to the existing models, our model has significant advantages in terms of parameters and computational complexity. However, in terms of detection performance, although our model does not achieve the highest accuracy, it performs excellently in meeting the real-time and lightweight requirements of lane detection on resource-constrained edge devices. Our method maintains a good balance between accuracy and real-time performance. Compared to existing state-of-the-art (SOTA) models, our model exhibits significant advantages in real-time performance.

Of course, our model still faces some challenges. For example, our model’s accuracy may be lower than existing state-of-the-art (SOTA) models, and we have not made any optimization improvements in this aspect. We have not made any changes to the backbone network part of the model. Therefore, in future work, we intend to use the ResNet backbone network of the model as a starting point to design an improved model. Based on this method, we aim to further enhance the model’s detection accuracy by designing new architectures and methods for the backbone network.

**Author Contributions:** Conceptualization was conducted by L.Y.; Data curation was conducted by G.Z.; Form analysis was performed by S.N. and Methodology was also developed by S.N.; S.L. conducted Project administration and Supervision; S.N. undertook original draft writing; Review and editing were performed by S.L. All authors have read and agreed to the published version of the manuscript.
**Funding:** This work was supported by the National Natural Science Foundation of China (61762085), and the Natural Science Foundation of Xinjiang Uygur Autonomous Region Project (2019D01C081): 61762085 and 2019D01C081

**Data Availability Statement:** The data are available upon request.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.