Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/95/2024CH0060

UFLD enhancements: New loss function and image masking

Yanbing Chen^{1,3,*,†}**, Qilong Wu**^{1,4,†}**, Xinyi Su**^{2,5}

^{1}Computer Science, Macau University of Science and Technology, Macau, 999078, China
^{1}Jinan University and University of Birmingham United College, Jinan University, Guangzhou, 510400, China
^{2}Computer Science and Technology, Chengdu University of Information Science and Technology, Chengdu, 610225, China

^{3}717172621@qq.com
^{4}1678090164@qq.com
^{5}1370700871@qq.com
*corresponding author
†These authors should be considered as co-first authors.

Abstract. In the field of autonomous driving, lane detection plays a crucial role in ensuring safety and accuracy in navigation. This paper presents enhancements to the Ultra Fast Lane Detection (UFLD) model by introducing a new loss function that ensures smoother and more continuous lane lines, and incorporating image masking techniques to simulate real-world occlusions. These modifications improve the model's robustness, particularly in scenarios with degraded lane markings. Through extensive experimentation on the TuSimple dataset, we demonstrate that the proposed enhancements lead to more accurate and realistic lane detection, especially in challenging environments.

Keywords: Lane detection, Autonomous driving, Image processing, Ultra Fast Lane Detection (UFLD)

1. Overview

In the rapidly advancing field of autonomous driving, ensuring the accuracy and reliability of lane detection systems is crucial. Current models, such as the Ultra Fast Lane Detection (UFLD) model [3], though effective, often struggle with complex driving conditions where lane markings are partially obscured or degraded. Additionally, maintaining the smoothness of detected lane curves is vital to ensure our realistic and safe vehicle navigation. Addressing these challenges is essential to enhance the robustness and precision of lane detection systems, thus improving overall driving safety.

Previous researches have extensively explored various lane detection techniques, including both traditional computer vision methods [1] and deep learning methods [2–6]. While traditional methods and advanced models have demonstrated significant improvements on real-time detection, they still have limits when applied to new and unseen scenarios, particularly where lane markings are partially obscured or missing.

Furthermore, there are several technical gaps still need to be addressed. Firstly, under complex road conditions such as poor lighting, glare, and blurred or missing lane markings, existing lane detection models still struggle to maintain high accuracy. Secondly, for practical application in autonomous driving systems, lane detection models need to ensure real-time processing capabilities, efficient computational performance, and high accuracy altogether. Additionally, current models lack adaptability and robustness when facing diverse road scenarios, such as urban roads, highways, rural roads and so on. Finally, the stability and accuracy of lane detection are quite significant, but still waiting for a better performance under different weather conditions, like rain and snow, and at different times of the day (daytime, nighttime).

In that case, our approach focusing on the issue of model's disappointing performance especially on the scenarios missing land marks, enhances the UFLD model by integrating a new loss function to restrict the smoothness of detected lane curves and introducing a random masking technique during the training phase, where segments of input images are intentionally obscured to simulate real-world occlusions. This combined approach aims to enhance the model's robustness, accuracy, and realism in diverse and dynamic driving environments.

We propose two techniques to address this issue:

Implementing Random Masking

For training purposes, we randomly mask the training images and make our training environment closer to the real environment. This can help our model learn more useful features.

Developing a New Loss Function

We incorporate a new loss function into the original ones. This could help with the improvement of lines' smoothness.

In summary, to make our model attain superior performance in challenging conditions we include random masking and construct a new loss function in our method. Our final model successfully enhances the trained model's robustness, accuracy, and smooth lane curves.

2. Related work

The fundamental challenge in our work is how to optimize some techniques within the framework of UFLD model. We compare traditional methods with deep learning approaches in our work. Traditionally, the field was dominated by conventional computer vision methods. They may implemented the traditional techniques, such as Sobel edge detection and Hough transform [1], to find track lines. While their methodology demonstrated some advantages in real-time scenarios, it was not robust for changing lighting or complex backgrounds.

In recent years, deep learning algorithms, especially convolutional neural networks, have greatly catch people's attention. Key technologies, such as real-time detection, robustness, and accuracy, have significantly advanced. For instance, LaneNet [5] achieved strong generalization performance when tested in complex driving conditions. Nonetheless, the model's success was based on labeled training data, which also to some extent contributes to our idea of randomly mask. Similarly, Pan et al. [2] proposed the SCNN, a network structure that processes images according to their spatial relationship order. While this method significantly improved the accuracy of lane structures by considering spatial dependencies, its computation cost limited its use in applications that require real-time implementations. Additionally, Neven et al. [4] advanced lane geometry estimation by designing an end-to-end deep neural network framework and using differentiable least-squares fitting to support lane line coordinate output. Although it filled a gap in older approaches, conventional real-time computation requirements limited this opportunity, which means poor real-time detection.

Among the advantages mentioned, Qin et al. explored Ultra Fast Lane Detection (UFLD) [3]. UFLD uses a lightweight backbone network following a row-based method to efficiently identify lanes. However, it is still lack evidence of an excellent effectiveness in complex situations. In addition, contrastive learning has emerged as a strong baseline for self-supervised lane detection. Zoljodi et al. [6] proposed Contrastive Learning for Lane Detection via Cross-Similarity (CLLD), which helps model adapt to real conditions with low visibility. This approach makes lane detection models more robust, but still it requires large amounts of data.

Based on these studies, we aim to enhance a model's efficiency in all kinds of scenarios. Consequently, we improve the UFLD model through random masking, which acts as a regularization strategy. The presence of such a technique during training helps increase the ability for self-supervised learning, which brings better robustness in closed-world conditions and environmental variations. Furthermore, we present a novel loss function specifically designed to smooth the lanes curved by the dots. In this way, we make our method perform better in diverse scenarios.

3. Methodology

In this part, we will adopt a new loss function and image processing method to improve the existing UFLD model, which is used to improve the model's lane line simulation under the condition of shelter and prevent the situation of lane line severance.

3.1. Loss function

In order to prevent the breaking of the simulated lane lines, we introduce a new loss function. This loss function consists of basic loss and smooth curve loss.

Base Loss: This basic loss function is the result of the loss function of the original UFLD model, and we found the corresponding value in the final calculation of the result and added our new function.

Smooth Curve Loss: The addition of a smooth curve forces the new model to maintain a continuous straight line or smooth curve when generating the simulated lane lines. In the study of the previous UFLD model, our team found that there was a system that generated broken line segments that did not belong to the part of the lane line. In order to prevent this situation, we added the new smooth curve loss function.

The total loss for SmoothCurveLoss is calculated as follows:

$$\text{SmoothCurveLoss} = \text{weight} \times \left(\frac{1}{B(N-2)} \sum_{b=1}^{B} \sum_{i=1}^{N-2} \|\mathbf{c}_i\|^2 + \frac{1}{B(N-1)} \sum_{b=1}^{B} \sum_{i=1}^{N-1} \|\mathbf{d}_i\|^2 \right)$$

where:

$$\mathbf{d}_i = P_{i+1} - P_i$$
$$\mathbf{c}_i = \mathbf{d}_{i+1} - \mathbf{d}_i$$

Detailed Formulas:

Assume points is a batch input of size B, and each input contains N points, denoted as :

$$P = \{P_1, P_2, \ldots, P_n\}$$

Calculate Differences:

The difference represents the difference between neighboring points:

$$d_i = P_{i+1} - P_i$$

Calculate Curvature:

The curvature represents the difference in differences of neighboring points:

$$c_i = d_{i+1} - d_i$$

Curvature Loss:

The curvature loss is the mean of the squares of all curvatures:

$$\text{curvature\_loss} = \frac{1}{B(N-2)} \sum_{b=1}^{B} \sum_{i=1}^{N-2} \|\mathbf{c}_i\|^2$$

Line Loss:
The line loss is the mean of the squares of all differences:

$$\text{line\_loss} = \frac{1}{B(N-1)} \sum_{b=1}^{B} \sum_{i=1}^{N-1} \|\mathbf{d}_i\|^2$$

Total Loss: The total loss function here combines the previous base loss and the smoothed curve loss, and adds their respective weights to it. The weight of the smooth curve function is only 0.5 because this function only acts as an auxiliary limit to a curve, and is not an implementation of the main function.
Combined Total Loss:
The combined total loss is calculated as follows:

$$\text{total\_loss} = 1.0 \times \text{base\_loss\_value} + 0.5 \times \text{smooth\_curve\_loss\_value}$$

3.2. Evaluation metrics

The model's performance is evaluated using the following metrics:
Accuracy: Measures the proportion of correctly detected lanes out of the total lanes.
Precision: Measures the proportion of true positive lane detections out of all positive detections.
Recall: Recall rate is an important evaluation index in the computer field, which is used to measure the ability of a model to identify positive examples. In other words, it is the proportion of positive cases identified by the algorithm to all actual positive cases.
F1: F1 is a harmonic average of accuracy and recall, which provides a comprehensive measure of the model's performance. It is especially important when dealing with unbalanced datasets because it takes into account both the accuracy and recall rates of the model, avoiding a single priority between the accuracy and recall rates.

4. Experimental Results

4.1. Data

4.1.1. Dataset Loading The dataset consists of lane detection images and corresponding ground truth labels. Images are sourced from the Tusimple dataset paths specified during setup.

4.1.2. Data Transformation The original images and labels are transformed as needed for our lane detection training, including steps like resizing, normalization, and augmentation.

4.1.3. Masking and Occlusion Simulation In real scenarios, there are always some lanes temporarily covered. To get close to this real-world situation, we implement the random mask.
These data processes are designed to train and test the UFLD model using a well-prepared dataset that reflects different real-world driving scenarios. These steps make the model more robust and reliable in lane detection.

4.2. Experiment setup

All of the operations will be done in cmd.

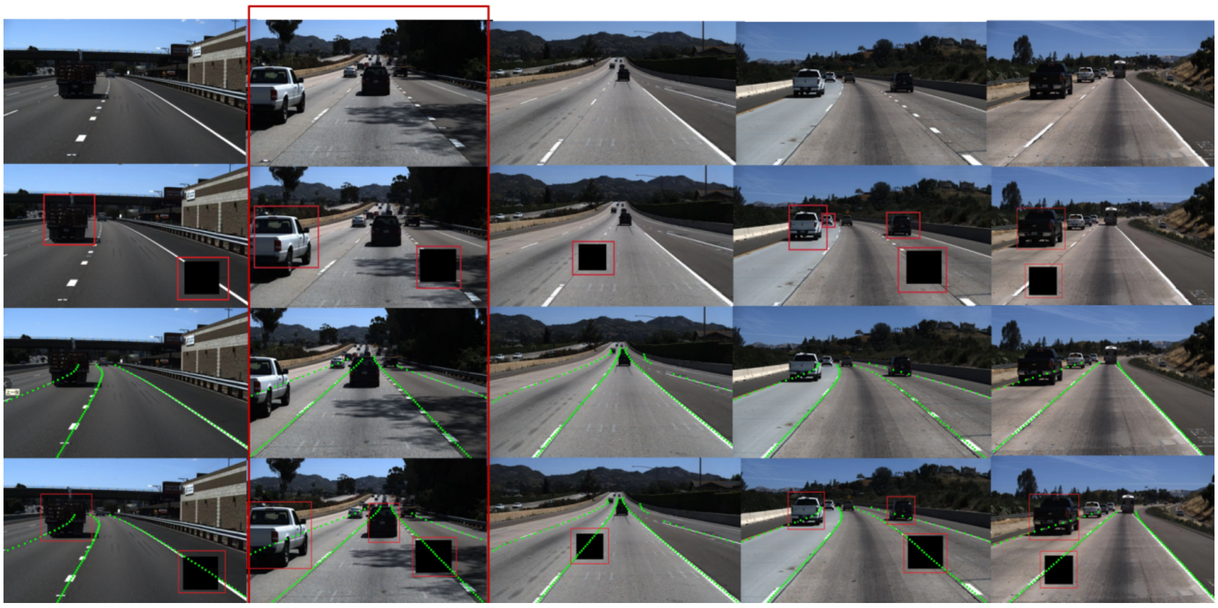


Figure 1. Image output after mask processing

4.2.1. Deploy the project First, get the relevant program and environment configuration from GitHub:

```
git clone https://github.com/cfzd/Ultra-Fast-Lane-Detection
cd Ultra-Fast-Lane-Detection
```

Next, create and activate a conda environment:

```
conda create -n lane-det python=3.11.9 -y
conda activate lane
```

Then, install dependencies:

```
conda install pytorch torchvision cudatoolkit=12.5 -c pytorch
pip install -r requirements.txt
```

Finally, prepare the data named TuSimple from Kaggle and extract it to TUSIMPLEROOT. The directory structure of TuSimple should look like Figure 2:

For Tusimple, the segmentation annotation is not provided, hence we need to generate segmentation from the json annotation.

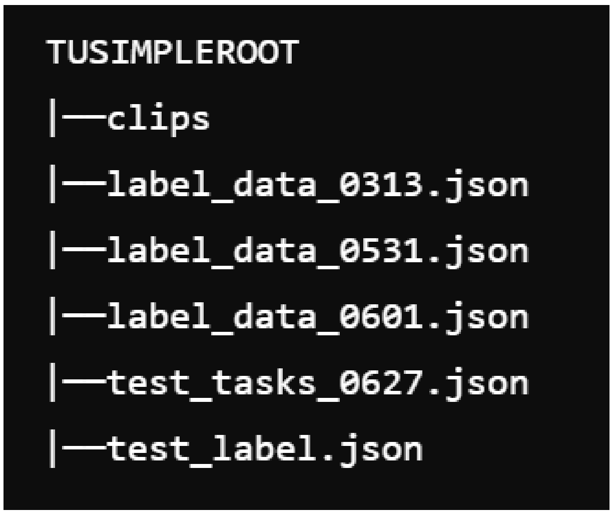
```
python scripts/convert\_tusimple.py --root TUSIMPLEROOT
```

4.2.2. Modify the directory and the corresponding index The parameter are modified in folder configs and we only use tusimple.py as we only use the dataset include the TuSimple, not have Culane.

The training parameters are modified in the train.py such as the folder which used to save the model named Log.

4.2.3. Conduct training We can use the following Command-line code to start train,

```
python train.py configs/tusimple.py
```

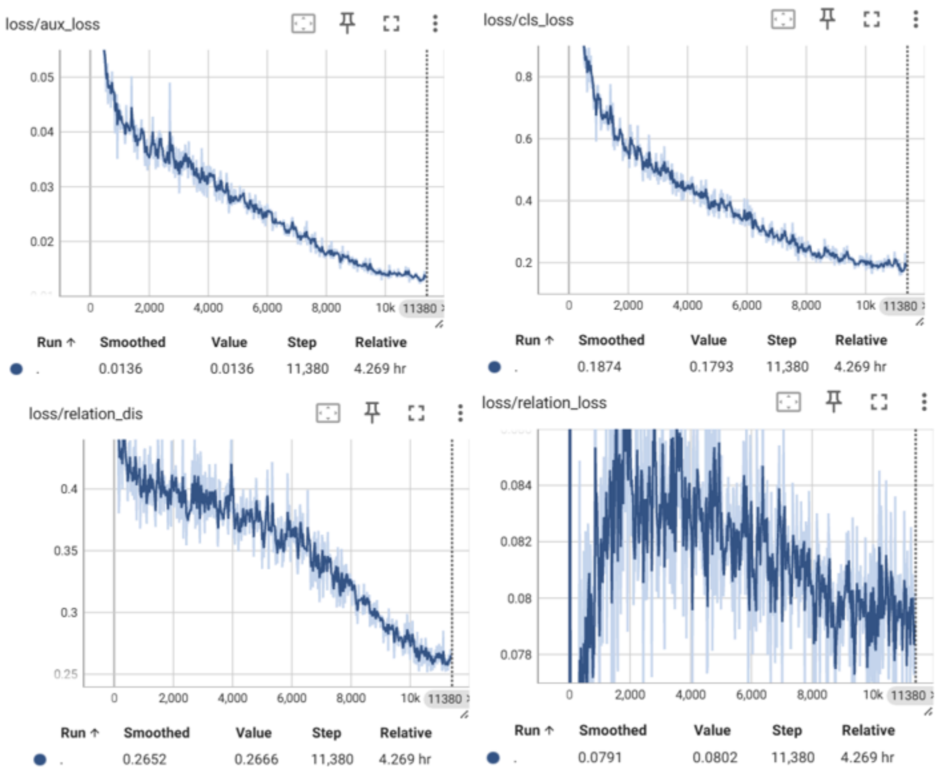


```
TUSIMPLEROOT
|—clips
|—label_data_0313.json
|—label_data_0531.json
|—label_data_0601.json
|—test_tasks_0627.json
|—test_label.json
```

Figure 2. Internal structure of the directory after data is downloaded

4.3. Experimental result

After 11380 iterations after 4.26 hours, the various loss function indicators are shown in the figure on the right.We can clearly see that the aux loss and cls loss both are rapidly decreasing until it is close to 0, gradually slowing down the rate of decline and close to the convergence.



loss/aux_loss

Run ↑	Smoothed	Value	Step	Relative
● .	0.0136	0.0136	11,380	4.269 hr

loss/cls_loss

Run ↑	Smoothed	Value	Step	Relative
● .	0.1874	0.1793	11,380	4.269 hr

loss/relation_dis

Run ↑	Smoothed	Value	Step	Relative
● .	0.2652	0.2666	11,380	4.269 hr

loss/relation_loss

Run ↑	Smoothed	Value	Step	Relative
● .	0.0791	0.0802	11,380	4.269 hr

Figure 3. Visualization of the results of four criteria(Accuracy, Precision, Recall, F1-score) over time and over the number of trials

It is clear that the value of the loss function is constantly oscillating around 0.3 and trying to become stable.

And we need to calculate some indicators to improve the model are becoming better.Then we can get the indicators like accuracy, precision, Recall and F1-score in p4.

Table 1. Results table after 100 experiments: Records the results and comparisons after the initial model and key times of training respectively

	tusimple_18	ep000	ep010	ep050	ep100
N	348	348	348	348	348
FP	0.1904	0.7919	0.2212	0.1972	0.1901
FN	0.0391	0.7637	0.0770	0.0465	0.0387
Accuracy	95.82%	71.76%	93.72%	95.23%	95.78%
Precision	79.28%	0.05%	74.28%	78.23%	79.32%
Recall	94.90%	0.03%	89.24%	93.84%	94.96%
F1-score	86.39%	0.00%	81.08%	85.33%	86.44%

Tusimple_18 is a less-trained reference model provided by the community. The ‘ep***‘ notation indicates that the model has been trained locally for ‘***‘ generations, where one generation involves training 100 times.

As the number of training sessions increases, metrics such as accuracy, precision, recall, and F1-score are constantly rising and approaching the ideal situation. This suggests that, over time, all performance indicators are expected to improve.

The visual display of these metrics is shown in the figures below:

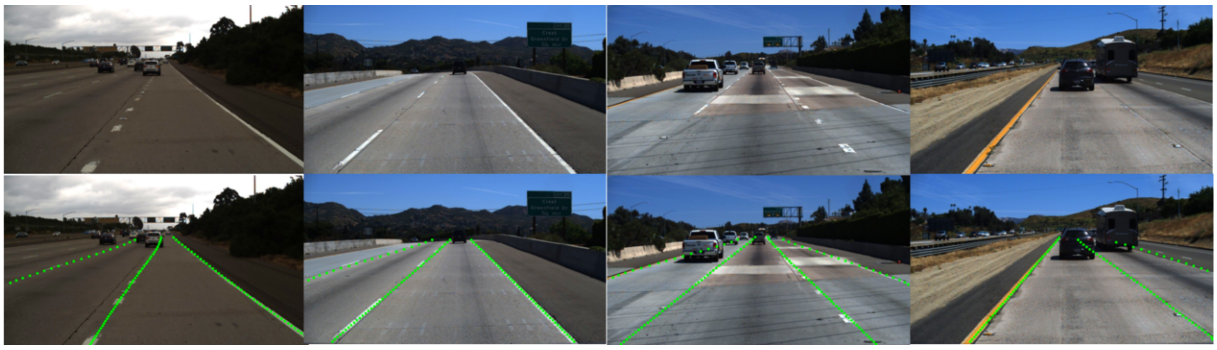


Figure 4. Model results after adding new image processing and loss functions

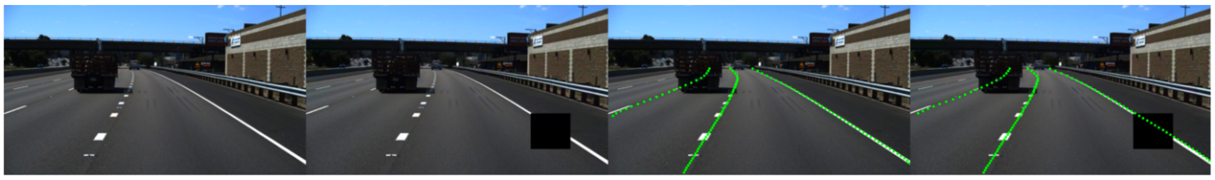


Figure 5. The same image is displayed after the graphics processing and the new loss function is added

5. Discussion and Future Work

5.1. Discussion

After adding loss function and image processing, we get a new UFLD model. After testing, the updated model showed a significant improvement in lane simulation results. However, this model still has its own shortcomings and use case limitations.

5.1.1. Limitations of the Current Model

Handling Intersections and Complex Road Segments

The main limitation of this model is the single use scenario. Because the new loss function limits the continuity of the simulated lane lines, the generated lane lines will be confused when facing intersections and other complex sections, resulting in errors when applied in the field of autonomous driving.

Dependency on Annotated Data

Another limitation of the model is its reliance on data sets that are already labeled. The data set we use is TuSimple. Although TuSimple is already a relatively comprehensive dataset containing many road conditions, it does not mean that TuSimple has covered all road types. Because the annotations in the data set are obviously extracted and identified in the model, it indicates that the model is dependent on these annotations. Therefore, this also limits the future generalization of the model to various road conditions and environments.

Sensitivity to Environmental Conditions

The performance of the model is also affected by weather and light. Even though we have added new image processing, the model can correctly recognize the obscured image. But conditions such as heavy rain, fog and low lighting can also pose significant challenges for models. These environments and light will affect the input of the image, resulting in difficulties in recognition, and then affect the accuracy of lane detection.

5.1.2. Future Work and Improvements

Adaptive Loss Functions

In order to solve the problem that the existing model cannot cope with intersections and complex road sections, an adaptive loss function can be added to judge when a continuous route is determined and when an intersection should be added in the subsequent implementation and update. In the face of complex situations can also be more flexible to judge. This increases the generality of the model.

Real-time Performance Optimization

In this stage of the experiment, we paid more attention to updating the results of the lane lines, so we did not change the real-time performance of the model. Although the UFLD model itself is designed to process images in real time, the subsequent research can further explore how to improve the speed of real-time calculation and reduce the complexity of calculation. This ensures that the updated model is more in line with the real-time performance of autonomous driving.

Evaluation on Diverse Datasets

For the annotation problem in the model-dependent data set, we can optimize the data set. For data sets that are already labeled, such as Tusimple, we can continue to use the current method of image extraction annotation. For some data sets that are not labeled in advance, we either label the images or develop new methods to locate the images and integrate them into the current UFLD model. This way our model does not rely solely on annotated data sets.

References

[1] J.W. Lee. A machine vision system for lane-departure detection. *Computer Vision and Image Understanding*, 86(1):52–78, 2002.

[2] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, New Orleans, LA, USA, February 2018.

[3] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV*, pages 276–291. Springer, 2020.

[4] 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.

[5] Z. Wang, W. Ren, and Q. Qiu. Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint*, 2018.

[6] A. Zoljodi, S. Abadijou, M. Alibeigi, and M. Daneshtalab. Contrastive learning for lane detection via cross-similarity. *Pattern Recognition Letters*, 185:175–183, 2024.