Lane Detection: A Semantic Segmentation Approach

Qiqi Wang, Fuen Chen, Xiaoming Liang

ABSTRACT

Driver assistance technology allows people not to concentrate on driving a car, in which the lane is the main and important features to guide vehicles and keep them safe. Traditional lane detection algorithms usually require high quality input images and hand-crafted features, that are computationally expensive and sensitive to environment. In order to overcome these problems, this paper comes up with an end-to-end lane detection method based on semantic segmentation, which has an accuracy as other deep learning methods, but has fewer parameters. In this paper, it uses two CNN blocks to extract features, and designs a loss function to help train the network. It has been tested on tuSimple dataset and could detect lanes in most of the images in the dataset.

KEYWORDS

Lane, Deep Learning, Semantic Segmentation.

INTRODUCTION

The development of self-driving car has led to a cascade of revolution in automotive industry. Now, whether traditional, Internet companies or startup companies are keeping up with self-driving technology. In driverless-car projects,

Qiqi Wang, Fuen Chen, Xiaoming Liang, School of Electronics and Information Engineering, Beijing Jiaotong University, No. 3 Shangyuancun, Haidian District, Beijing, China
sensing the surrounding environment is one of the most important parts. Lanes exist in most formal roads, and can be used to understand position of vehicles, which we are driving, to make proper decisions. Only when vehicles have an accurate and timely judgement of the car’s position related to lanes can it operate safely.

The most common sensor used for lane detection is a RGB camera which has a low price. However, only with pixel color information, it is hard to detect it accurately. For example, most traditional methods for lane detection focus only on low-level features, like color (e.g. [1], [2]), edge (e.g. [3], [4]), especial canny edge detection. These low-level features always come with Hough transform [5]. These methods are simple and can be applied in some conditions without too much extra modification, but the results are quite depending on the environment especially lighting condition.

When Alexnet came out in 2012, people found convolution neural network could do a great job in the field of computer vision especially in complex environments. CNN was applied in many recent methods for feature extract [6]. Image segmentation methods are used for lane detection recently like semantic segmentation [7] but with too many parameters.

For all these reasons and more, this paper broaches a lane detection method based on ERFNet with fewer parameters but high accuracy. CNN was used to extract features. The loss function of this network is inspired from PINet [8], which consider the confidence and position of the lane. Experimental results indicate that the network can detect the lane effectively.

**BASIC ERFNET**

Image segmentation is dividing up a given image into classes like the road, cars, sidewalk, buildings, etc. ERFNet is one of semantic segmentation techniques, which balances the accuracy and operating speed with the use of Non-bottleneck-1D layer. The network can learn the residual function in this way to help training and significantly reduce the computational cost.

Non-bottleneck-1D is a further development of Non-bottleneck and Bottleneck 9, which are describe by following formula:

\[
a_i^1 = \varphi \left( b_i^h + \sum_{l=1}^{L} \overline{H}_i^l \ast \left[ \varphi \left( b_i^v + \sum_{c=1}^{C} \overline{v}_{lc} \ast a_i^0 \right) \right] \right)
\]

Where, \( L \) is the quantity of filters in the intermediate layer, \( b \in \mathbb{R}^F \) is the vector representing the bias term for each filter, \( \overline{v}_{lc} \) and \( \overline{H}_i^l \) are the vectors of length \( d \),
d_h * d_v is the kernel size of each feature map (typically d_h ≡ d_v ≡ d), a^0 is its input, C is the total amount of input planes, φ(⋅) could be implemented with ReLU or PReLU[11]. It is showed as following figure:

Compared with the bottleneck design, the module is faster (same as the calculation time) and has fewer parameters, while maintaining the same learning ability and accuracy as the non-bottleneck design [11].

The module proposed above is used in segmentation project and get increased width as well as computationally efficient. The design of ERFNet is to stack the recommended non-bt-1D layers in order, so as to maximize its learning performance and efficiency [11]. This network structure is designed as the encoder-decoder architecture like SegNet [12] and ENet [13].

Experiments demonstrate that the network provides a great balance between reliability and speed, so we choose ERFNet as the base network for our research.

LANE-DETECTION-NETWORK

Network Structure

The entire structure basically follows ERFNet, which generates semantic segments end-to-end with the same resolution as inputs, but it resizes pictures into size 512*256 before training. The feature extraction layer is inspired from the stacked hourglass network. This paper uses two ERFNet blocks for the feature
extraction. Each block has two output branches, and output size is the same as the input size. Figure 2 is an architecture of the whole network.

![Figure 2. Network structure.](image)

The size of input images is 512*256, and it passes to feature extraction layer to extract feature and generate loss from both ERFNet blocks.

![Figure 3. ERFNet block.](image)
Figure 3 shows the detailed architecture about the ERFNet block. In Figure 3, blue boxes denote downsampler block, green boxes denote Non-bottleneck-1D (Non-bt-1D) layers, yellow boxes denote Non-bottleneck-1D block (Non-bt-1D block) layers, and orange boxes denote deconvolution layers. The detail of all layers mentioned above is showed at Table I.

**TABLE I. THE DETAIL OF ALL LAYERS.**

| Layer | Type             | out-F | out-Res |
|-------|------------------|-------|---------|
| 1     | Downsampler block| 16    | 256x128 |
| 2     | Downsampler block| 64    | 128x64  |
| 3-7   | 5+Non-bt-1D      | 64    | 128x64  |
| 8     | Downsampler block| 128   | 64x32   |
| 9-12  | Non-bt-1D block  | 128   | 64x32   |
| 13-16 | Non-bt-1D block  | 128   | 64x32   |
| 17    | Deconvolution (upsampling) | 64 | 128x64 |
| 18-19 | 2+Non-bt-1D      | 64    | 128x64  |
| 20    | Deconvolution (upsampling) | 16 | 256x128 |
| 21-22 | 2+Non-bt-1D      | 16    | 256x128 |
| 23    | Deconvolution (upsampling) | 3  | 512x256 |
| 9/13  | Non-bt-1D (dilated 2) | 128 | 128x64 |
| 10/14 | Non-bt-1D (dilated 4) | 128 | 128x64 |
| 11/15 | Non-bt-1D (dilated 8) | 128 | 128x64 |
| Non-bt-1D block 12/16 | Non-bt-1D (dilated 16) | 128 | 128x64 |

The Downsampler module is designed by connecting a single 3*3 convolutional parallel output with stride 2 and a Max-pooling. The Non-bt-1D block is four Non-bt-1D layers with different dilated convolutions (2, 4, 8, 16) to get more context.

**Loss Function**

**CONFIDENCE BRANCH**

The confidence branch expresses confidence values of each grid. It is passed to the next block, and it improves training stability. Equation 1 shows the loss function of the confidence branch.
\[ L_{\text{confidence}} = \frac{1}{N_e} \gamma_e \sum_{g \in G_e} (g^*_c - g_c)^2 + \frac{1}{N_n} \gamma_n \sum_{g \in G_n} (g^*_c - g_c)^2 \]  

(2)

Where \( N_e, N_n \) are the number of grids that a point exist or non-exist in, \( G \) represents a set of grids, \( g_c \) is a confidence output of the grid, \( g^*_c \) denotes the ground-truth, and \( r \) represents each coefficient.

OFFSET BRANCH

From the offset branch, we can acquire exact locations. The value of the offset branch output is between 0 and 1, and means a position related to a grid. In this paper, a grid is matched to 1 pixel depending on the ratio between input size and output size. There are two channels for predicting x-axis offset and y-axis offset. Equation 3. shows loss function of the offset branch

\[ L_{\text{offset}} = \frac{1}{N_e} \gamma_x \sum_{g \in G_e} (g^*_x - g_x)^2 + \frac{1}{N_e} \gamma_y \sum_{g \in G_e} (g^*_y - g_y)^2 \]  

(3)

TOTAL LOSS

The total loss \( L_{\text{total}} \) is the summation of the above two loss terms, and we use the following total loss for training of the entire network.

\[ L_{\text{total}} = a L_{\text{confidence}} + b L_{\text{offset}} \]  

(4)

In training step, we take all coefficients 1.0 initially, and add 0.5 to \( a \) and \( \gamma_n \) at last few epochs. The proposed loss function is adapted to the end of each ERFNet block, and it betters training stability for whole network.
EXPERIMENTAL RESULTS AND ANALYSIS

We use tuSimple [14] dataset to train the network and choose Adam optimizer. This paper use AWS EC2 p2.xlarge instance to train, which has 61 GiB memory and 4 vCPUs.

Evaluation Metrics

Accuracy is the main evaluation metric of the tuSimple dataset, and it is set by following equation, which means the average number of the correct points.

\[
\text{accuracy} = \sum_{\text{clip}} \frac{C_{\text{clip}}}{S_{\text{clip}}} 
\]

(5)

Where, \( C_{\text{clip}} \) is the number of the correct predicted points generated from the trained module, and \( S_{\text{clip}} \) is the number of ground-truth point in the corresponding image. False negative and false positive are also used to evaluate.

\[
FP = \frac{F_{\text{pred}}}{N_{\text{pred}}} 
\]

(6)

\[
FN = \frac{M_{\text{pred}}}{N_{\text{gt}}} 
\]

(7)

Where, \( F_{\text{pred}} \) is the number of wrongly predicted lanes, \( N_{\text{pred}} \) represents the number of lanes that the network predicts, \( M_{\text{pred}} \) denotes the total amount of missed lanes, and \( N_{\text{gt}} \) is the quantity of ground-truth lanes.
Experiments

There are 3626 annotated data for training and 2782 images for testing in tuSimple dataset. This paper applies some data augmentation methods like translation, rotation, flip, adding Gaussian noise, and changing intensity to diversify the dataset.

Figure shows the results in both straight and curved lanes, we can see it detect the lane effectively.

![Figure 4. Experiment Results.](image)

Table II shows the detailed evaluation results. This network gets quite good results among these lane detection methods. The accuracy has reached 96.61%.

| Method                  | Acc/% | FP    | FN    |
|------------------------|-------|-------|-------|
| SCNN[16]               | 96.53 | 0.0617| 0.018 |
| LaneNet(+H-net)[17]    | 96.38 | 0.078 | 0.0244|
| PointLaneNet(MobileNet)[18] | 96.34 | 0.0467| 0.0518|
| Enet-SAD[19]           | 96.64 | 0.0602| 0.0205|
| Ours                   | 96.61 | 0.0501| 0.0361|
Figure 5 shows the amount of the parameters in some methods, and it indicates that the proposed network is a lighter method among other ones. Because the use of Non-bt-1D saves a lot of memory.

![Figure 5. Model analysis: parameters (M).](image)

**CONCLUSION**

A lane detection method based on ERFNet is proposed, which comes with confidence and offset loss to stable training. The network gets an accuracy of 96.61%, uses fewer parameters, and can guide cars well on some roads.

However, most of the data we used are from standard roads, not all situations are so ideal, like Hutong or roads in communities where there are few lanes. Application, improvement, and evaluation of this module in more data are the focus of the next research.

**REFERENCES**

1. He, Yinghua, Hong Wang, and Bo Zhang. “Color-based road detection in urban traffic scenes.” IEEE Transactions on intelligent transportation systems 5.4 (2004): 309-318.

2. B Chiu, Kuo-Yu, and Sheng-Fuu Lin. “Lane detection using color-based segmentation.” IEEE Proceedings. Intelligent Vehicles Symposium, 2005. IEEE, 2005.

3. Wang, Yue, Dinggang Shen, and Eam Khwang Teoh. “Lane detection using catmull-rom spline.” IEEE International Conference on Intelligent Vehicles. Vol. 1. 1998.
4. Lee, Chanho, and Ji-Hyun Moon. “Robust lane detection and tracking for real-time applications.” IEEE Transactions on Intelligent Transportation Systems 19.12 (2018): 4043-4048.

5. Duda, R. O., and P. E. Hart. “Use of the Hough transform to detect lines and curves in pictures.” Commun. ACM, vol. 15, no. 1, pp. 1115, 1972.

6. Bojarski M, Del Testa D, Dworakowski D, et al. End to End Learning for Self-Driving Cars[J]. 2016.

7. Ze Wang, Weiqiang Ren, and Qiang Qiu. Lanenet: Real-time lane detection networks for autonomous driving, 2018.

8. Ko Y, Jun J, Ko D, et al. Key Points Estimation and Point Instance Segmentation Approach for Lane Detection[J]. 2020.

9. E. Romera, J. M. A´ lvarez, L. M. Bergasa, and R. Arroyo, “Efficient convnet for real-time semantic segmentation,” in IEEE Intelligent Vehicles Symp. (IV), 2017, pp. 1789–1794.

10. K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proceedings of the IEEE International Conf. on Computer Vision, 2015, pp. 1026–1034.

11. Romera E, Alvarez J M, Bergasa L M, et al. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation[J]. IEEE Transactions on Intelligent Transportation Systems, 2017.

12. V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling,” arXiv preprint arXiv:1505.07293, 2015.

13. A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “Enet: A deep neural network architecture for real-time semantic segmentation,” arXiv preprint arXiv:1606.02147, 2016.

14. The TuSimple lane challenge, http://benchmark.tusimple.ai/.

15. Pan, Xingang, et al. “Spatial as deep: Spatial cnn for traffic scene understanding.” Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

16. Neven, Davy, et al. “Towards end-to-end lane detection: an instance segmentation approach.” 2018 IEEE intelligent vehicles symposium (IV). IEEE, 2018.

17. Chen, Zhenpeng, Qianfei Liu, and Chenfan Lian. "PointLaneNet: Efficient end-to-end CNNs for Accurate Real-Time Lane Detection." 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2019.

18. Hou, Yuenan, et al. “Learning lightweight lane detection cnns by self attention distillation.” Proceedings of the IEEE International Conference on Computer Vision. 2019.