End-to-End Lane Marker Detection via Row-wise Classification

Seungwoo Yoo Hee Seok Lee Heesoo Myeong
Sungrack Yun Hyoungwoo Park Janghoon Cho Duck Hoon Kim

Qualcomm Korea YH
{yoos,heeseokl,hmyeong,sungrack,hwoopark,janghoon,duckhoon}@qti.qualcomm.com

Abstract

In autonomous driving, detecting reliable and accurate lane marker positions is a crucial yet challenging task. The conventional approaches for the lane marker detection problem perform a pixel-level dense prediction task followed by sophisticated post-processing that is inevitable since lane markers are typically represented by a collection of line segments without thickness. In this paper, we propose a method performing direct lane marker vertex prediction in an end-to-end manner, i.e., without any post-processing step that is required in the pixel-level dense prediction task. Specifically, we translate the lane marker detection problem into a row-wise classification task, which takes advantage of the innate shape of lane markers but, surprisingly, has not been explored well. In order to compactly extract sufficient information about lane markers which spread from the left to the right in an image, we devise a novel layer, inspired by [8], which is utilized to successively compress horizontal components so enables an end-to-end lane marker detection system where the final lane marker positions are simply obtained via argmax operations in testing time. Experimental results demonstrate the effectiveness of the proposed method, which is on par or outperforms the state-of-the-art methods on two popular lane marker detection benchmarks, i.e., TuSimple and CULane.

1. Introduction

With the explosive growth of the researches and developments on the computer vision technologies with sensor fusion, localization and path planning, the advanced driver assistance system (ADAS) or high-level self-driving system (SDS) has been widely adopted in recent vehicles such as Waymo [31], Uber [1], Lyft [2], Mobileye [34], Google car [27] and Tesla [6]. Especially, recent researches and projects [9, 6, 34] on the ADAS and SDS are focused more on cameras than other sensors, *e.g.* LiDAR, due to the cost, design, and also big accuracy improvements in the camera-based perception systems. Although there are a number of components related to the ADAS or SDS, such as lane marker detection, vehicle detection & tracking, obstacle detection, scene understanding, and semantic segmentation, lane marker detection is one of the key components in camera perception and positioning for several applications, *e.g.*, lane keeping/change assist.

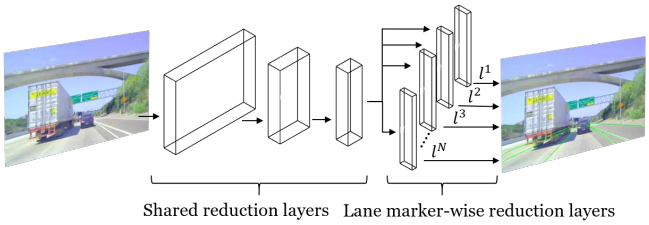


Figure 1. The **E2E-LMD** framework for lane marker detection.

A number of researches on lane marker detection have been proposed [25, 13, 19, 8, 18, 10, 22, 26, 29, 7, 5, 17, 14]. Most conventional lane marker detection methods are based on two-stage semantic segmentation approaches [23, 16, 12]. In the first stage of these approaches, a network is designed to perform a pixel-level classification to assign each pixel in an image to the binary label, *i.e.*, lane marker or not. However, in each pixel classification, the dependencies or structures between pixels are not specifically considered, and thus additional post-processing is performed in the second stage to explicitly impose the constraints such as uniqueness or straightness of the detected lane marker. The post-processing can be implemented with conditional random field, additional networks, or sophisticated CV techniques like RANSAC, but its computational complexity is not negligible and it should be carefully combined with the first stage by hand-tuning. Therefore, there approaches are hard to scale up for various environments and datasets. Another lane marker detection methods are generative adversarial network (GAN)-based approaches [19, 10, 21] which considers additional loss to impose such structural constraints.

In this paper, we consider a simple end-to-end framework for recognizing lane marker, called *E2E-LMD*, which directly predicts the lane marker vertices without any sophisticated post-processing step (Figure 1). Here, the lane marker recognition problem is considered as multiple row-wise classification tasks for each lane marker type where features for classification are expressed through a two-stage module, and the final lane marker positions are simply obtained by *argmax* operations in testing time. The first-stage layers successively compress and model the horizontal components for the shared representation of all lane markers, and the second-stage layers separately model the each lane marker based on this shared representation to directly output the lane marker vertices.

In summary, the contribution of this paper can be summarized as follows: 1) We present a novel and intuitive framework for detecting lane markers. 2) The proposed method is on par or outperforms the recent state-of-the-art methods in both benchmark datasets, *i.e.*, TuSimple and CULane, without complex post-processing. And, finally, 3) We show that the proposed method can effectively capture lane marker representation in an efficient manner with extensive experiments and visualization.

2. Related Work

Most traditional lane marker detection methods are based on hand-crafted low-level features. In [3], they proposed the line segment detection using selective Gaussian spatial filters, which is followed by post-processing steps. Recently, deep learning-based methods are employed to learn to extract features at various scenes. There are mainly two approaches based on convolutional neural networks (CNN): 1) Segmentation-based approach and 2) GAN-based approach.

The first approaches consider lane marker detection as a semantic segmentation task [25, 22, 7, 14, 13]. In [22], the benefits of lane marker segmentation are combined with a clustering approach designed for instance segmentation. In [25], they train a spatial CNN (SCNN) with propagating message as residual for detecting long continuous structure. In [14], pixel-wise clustering is applied based on conventional segmentation network. In [7], authors proposed a deep neural network that predicts a weights map like a segmentation output for each lane marker and a differentiable least-squares fitting module for mapping parameters for curve fitting. In [13], self-attention distillation (SAD) is proposed to allow the network to exploit attention maps within the network itself and complements the segmentation-based supervised learning.

Second, some methods adopt GAN for lane marker detection tasks. In [10], authors take lane marker labels as extra inputs and use GAN so that the segmentation maps resemble labels to predict the better segmentation outcomes. In [19], they generate low light conditioned images using GAN to increase the environmental adaptability of the network.

Other deep learning-based methods make an effort to solve lane marker detection from different aspects. In [17], they use extra labels of vanishing point to train the network to output better structural information. In [5], they consider the lane marker detection and classification problems as regression problems.

One work close to the proposed method is [8, 18] where column-wise representation is used to recognize free space in road scenes. This horizontal representation for detecting obstacles has been easily utilized for autonomous driving tasks since it can be efficiently translated to an occupancy grid representation. Based on the representation, they used convolutional neural network with simple successive vertical pool layers to regress free space boundaries.

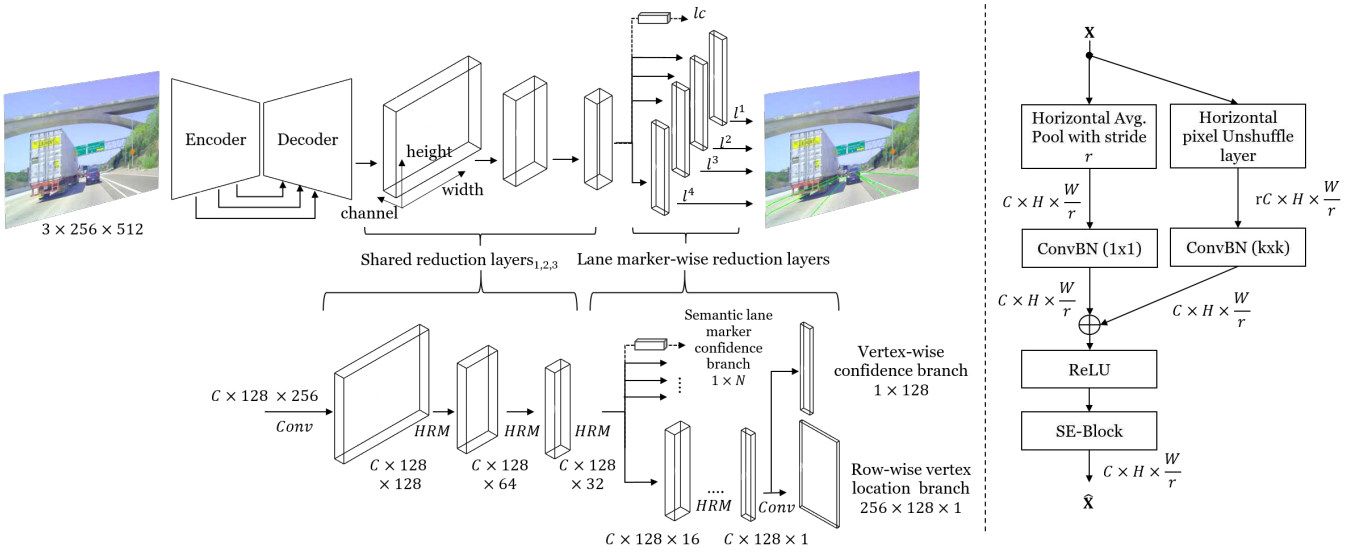
3. Proposed Method

As reviewed in Sec. 2, the lane marker detection problem has been tackled with various approaches and each of them has its own pros and cons. However, most of them are based on semantic segmentation with complex post-processing which hinders end-to-end training for extracting lane marker positions. Inspired by recent works [8, 18], we consider the above problem as finding the set of horizontal locations of each lane marker in an image. Specifically, we divide an image into rows and obtain a row-wise representation for each lane marker using a convolutional neural network. Then lane marker detection can be thought as row-wise classification. In other words, contrasted to the conventional segmentation-based lane marker detection, the proposed method can directly provide lane marker positions. More specifically, given an input image $\mathbf{X} \in R^{3\times h\times w}$ where h and w are the image height and width, respectively, the objective is to find a lane marker l_i $(i = 1, \cdots, N)$ represented by the set of vertices $\{vl_{ij}\} = \{(x_{ij}, y_{ij})\}$ $(j = 1, \cdots K)$. Here, N is the number of lane markers in $\mathbf{X}$ which is generally pre-defined, and K is the total number of vertices that is limited to h due to the row-wise representation.

The details of the proposed architecture, which is conceptually simple and can be utilized to any segmentation-based approaches, and its training and inference will be described in the following subsections.

3.1. Network Architecture

Architecture Design: We propose a novel architecture composed of successive shared and lane marker-wise horizontal reduction modules (HRMs), which leads to removing horizontal components spatially and setting the channel size as the target width resolution.



(a) The schema of the **E2E-LMD** (b) The horizontal reduction module (HRM)

Figure 2. The **E2E-LMD** architecture for lane marker detection. We extend general encoder-decoder architectures by adding successive horizontal reduction modules for end-to-end lane marker detection. Numbers under each block denote spatial resolution and channels. **(a)** Arrows with HRM denote a horizontal reduction module of (b). Arrows with *Conv* are output convolution with 1×1. Dashed arrows denote the global average pooling with a fully connected layer. **(b)** HRM is utilized to compress the horizontal representation. r denotes the pooling ratio for width part. Conv kernel size k is set as 3 except the last HRM layer which set as 1.

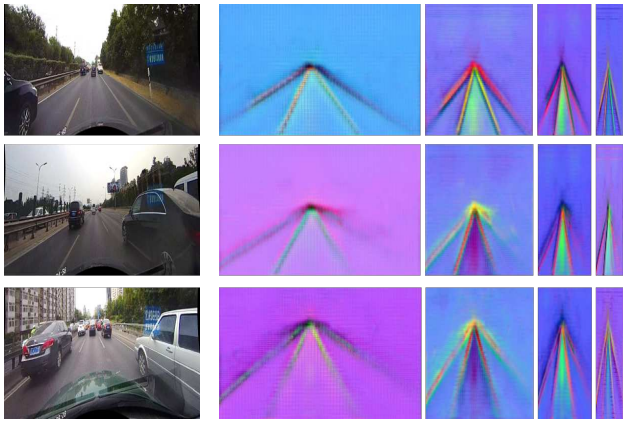
The proposed end-to-end lane marker detection (**E2E-LMD**) architecture consists of three stages (see Fig. 2(a)). The first stage is a general encoder-decoder segmentation network [30] which encodes information of lane markers in an image and reconstructs spatial resolution. In contrast to standard semantic segmentation approaches, in our implementation, we only recover spatial resolution as the half of an input size to reduce computational complexity.

In the second stage, we successively squeeze the horizontal dimension of the shared representation using HRMs without changing the vertical dimension. With this squeeze operation, we can obtain the row-wise representation in a more natural way. After running shared HRMs, we squeeze the remaining width of representation by lane marker-wise HRMs to make single vector representation for each row. We found that it is required to assign dedicated HRMs on each lane marker after the shared HRMs for increasing accuracy numbers, since each lane marker has different innate spatial and shape characteristics. For computational efficiency, however, only the first few HRMs are shared across lane markers, followed by lane marker-wise HRMs. With more shared layers we can save computational cost but each lane marker accuracy might be degraded.

In the last third stage, we have two branches for a lane marker l_i: a row-wise vertex location branch and a vertex-wise confidence branch. These branches perform classification and confidence regression on the last HRMsfeatures where spatial resolution only has the vertical dimension while the channel size meets the target horizontal resolution h', *i.e.*, $h' = h/2$. The row-wise vertex location branch predicts the horizontal position x_{ij} of l_i per y_{ij} $(y_{ij} = 0, \cdots, h')$.

The vertex-wise confidence branch predicts the existence confidence vc_{ij} whether (x_{ij}, y_{ij}) is valid or not. Following [25], we also add a semantic lane marker confidence branch which produces lane marker-wise existance confidence lc_i after shared HRMs.

Horizontal Reduction Module: To effectively compress the horizontal representation, we utilize residual layers proposed in [11] (see Fig. 2(b)). Specifically, in the skip connection, we add a horizontal average pooling layer with a 1×1 convolution to down-sample horizontal components. Although pooling operations let the deeper layers gather more spatial context (to improve classification) and reduce computational complexity, they still have the drawback of reducing the pixel precision. Therefore, to effectively keep and enhance the horizontal representation, inspired by the pixel shuffle layer of [32, 24], we propose to rearrange the elements of $C \times H \times W$ input tensor to make a tensor of shape $rC \times H \times W/r$ in the residual branch, which is somewhat a reverse operation of the original pixel shuffle block in [32] so called the horizontal pixel unshuffle layer. By rearranging the representation, we can efficiently move spatial information to channel. Then we apply a convolution operation to reduce the increased channel rC to C which not only reduces computational complexity but also helps to ef-



Input Learned representations in successive layers

Figure 3. **Learned representations on decoder and shared HRM layers$_{1,2,3}$:** We visualize how features are encoded in different depths of our shared HRM layers after decoder. For each layer (row), we visualize the first three principal components as RGB values at each spatial locations. We observe that the features become more distinctive, adapted to specific locations and disentangled in the later layers.

fectively compress lane marker spatial information from the pixel unshuffle operation.

To further improve the discrimination between lane markers, we add an attention mechanism by adding Squeeze and Excitation (SE) block [15]. The SE block helps to include global information in the decision process by aggregating the information in the entire receptive field and recalibrates channel-wise feature responses which have spatial information encoded by the horizontal pixel unshuffle layer (see Fig. 3 and Fig. 4).

To confirm the effectiveness of the proposed architecture, we visualize the learned representation using PCA (Principal Component Analysis) (see Fig. 3). The visualized results show that the proposed architecture successfully compress the spatial lane marker information even though we squeeze the horizontal components in representations.

3.2. Training

The training objective is to optimize total loss L given by

$$L = L_{vl} + \lambda_1 L_{vc} + \lambda_2 L_{lc}, \tag{1}$$

where L_{vl}, L_{vc}, and L_{lc} are losses for lane marker vertex location, lane marker vertex confidence, and lane marker-wise confidence, respectively. And λ_1 and λ_2 are weights for the last two losses.

Lane Marker Vertex Location Loss: As we formulated lane marker detection as row-wise classification on lane marker's horizontal position, any loss function for classification can be used.

Specifically, we tested three loss functions, *i.e.*, cross-entropy (CE), KL-divergence (KL), and PL-loss (PL) [18]. The CE loss L_{ij}^{CE} for lane marker l_i at a vertical position y_{ij} is computed using the ground truth location x_{ij}^{gt} and the predicted logits f_{ij} having $W/2$ channels.

To train the lane marker vertex location branch using the KL loss L_{ij}^{KL}, we first make a sharply-peaked target distribution of lane marker positions as a Laplace distribution $Laplace_{gt}(\mu, b)$ with $\mu = x_{ij}^{gt}$ and $b = 1$, and then compare it with an estimated distribution $Laplace_{pred}(\mu, b)$ by

$$\begin{aligned} \mu &= \mathbb{E}_{f_{ij}}[x_{ji}] \\ &= softargmax(x_{ji}) = \sum_{W/2} softmax(f_{ij}) \cdot x_{ij} \\ b &= \mathbb{E}_{f_{ij}}[|x_{ji} - \mathbb{E}_{f_{ij}}[x_{ji}]|] \end{aligned} \tag{2}$$

, similarly with the 2D facial landmark detection algorithm in [28, 4]. In case of the PL loss, we follow the original formulation of [18] by modeling the probability of lane marker positions as piecewise linear probability distribution.

For an input image, the total lane marker vertex location loss is given by

$$L_{vl} = \frac{1}{N} \sum_{i}^{N} \frac{1}{\sum_{j}^{K} e_{ij}} \sum_{j}^{K} L_{ij}^{type} \times e_{ij} \tag{3}$$

, where $type \in \{CE, KL, PL\}$, e_{ij} denotes whether ground truth exists or not, *i.e.*, $e_{ij} = 1$ if there is l_i having a lane marker vertex at y_{ij} and $e_{ij} = 0$ if not.

Lane Marker Vertex Confidence Loss: The lane marker vertex existence is a binary classification problem, thus it can be trained using a binary CE loss L_{ij}^{BCE} between single scalar-value prediction at each y_{ij} location of lane marker l_i and ground truth existence e_{ij}. The loss for an entire image is then computed as $L_{ve} = \frac{1}{N \times K} \sum_{i}^{N} \sum_{j}^{K} L_{ij}^{BCE}$.

Lane Marker Label Loss: Following [25], we add a binary CE loss L_{i}^{BCE} to train the lane marker-wise existence prediction. The loss is computed using the predicted N-dimensional vector lc_i and existence of each lane l_i in the ground truth. The total loss is then computed as $L_{le} = \frac{1}{N} \sum_{i}^{N} L_{i}^{BCE}$.

3.3. Inference

In testing time, lane marker vertices can be simply estimated per loss as follows: the *argmax* operation is used for the CE or PL loss, and the *softargmax* operation is used for the KL loss. As mentioned above, there are three outputs from the proposed architecture, *i.e.*, horizontal location of lane marker vertices x_{ij}, vertex-wise existence confidence vc_{ij}, and lane marker-wise existence confidence lc_i. Then

Input Before SE After SE

Figure 4. **Learned representations at shared HRM layers$_{2,3}$ before/after SE module:** We visualize how encoded features are changed before/after SE block. We observe that after SE block, lane representations become more discernible to be easily separate from each other.

(a) TuSimple (b) CULane

Figure 5. The examples of video frames of (a) TuSimple [33] and (b) CULane [25]. Ground truth lane markers are shown in various colored lines.

the final lane marker vl_{ij} for l_i can be obtained by

$$\{vl_{ij}\} = \begin{cases} \{(x_{ij}, y_{ij}) | vc_{ij} > T_{vc}\} & \text{if } lc_i > T_{lc}, \\ \emptyset & \text{else,} \end{cases} \quad (4)$$

where T_{vc} and T_{lc} are the thresholds of vertex-wise existence confidence and lane marker-wise existence confidence, respectively. Specifically, the sigmoid output of the vertex-wise and lane marker-wise existence branches is utilized to reject low-confident lane marker vertices and lane markers, respectively.

4. Experiments

Datasets: We consider two lane marking datasets for evaluating our method. TuSimple [33] and CULane [25] are widely used in previous works. Some examples of these datasets with ground truth are shown in Fig. 5.

1) *TuSimple*. The TuSimple dataset consists of 6,408 road images on US highways. The resolution of image is 1280×720. The dataset is composed of 3,626 for training, 358 for validation, and 2,782 for testing called the TuSimple test set of which the images are under different weather conditions.

2) *CULane*. The CULane dataset consists of 55 hours of videos which comprise urban, rural and highway scenes, and 133,235 frames are extracted from videos. The dataset is divided into 88,880 frames for training, 9,675 for validation, and 34,680 for testing called the CULane test set. The images have a resolution of 1640×590. The test set contains 9 different challenging driving scenarios ("Normal", "Crowd", "Highlight", "Shadow", "Arrow", "Curve", "Cross", "Night" and "No line").

Evaluation Metrics: For comparing the proposed method with previous lane marker detection methods, we used the following evaluation metrics for each particular dataset:

1) *TuSimple*. We report the official metric used in [33] as the evaluation criterion. The accuracy is calculated as the average correct number of vertices per image: $Accuracy = \frac{N_{correct}}{N_{gt}}$, where $N_{correct}$ is the number of correctly predicted lane marker vertices, and N_{gt} is the number of ground truth lane marker vertices. Also, we report the false positive (FP) and false negative (FN) scores.

2) *CULane*. As in [25], for judging whether the proposed method detects lane markers correctly, we consider each lane marking as a line with 30 pixel width and compute the intersection-over-union (IoU) between ground truths and predictions. Predictions whose IoUs are larger than 0.5 are considered as true positives (TP). Then, we used F_1-measure as the evaluation metric, which is defined as: $F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

Implementation Details: We resized the image of TuSimple and CULane to 256×512 and set N as 6 and 4 for each dataset. To assign an unique class ID to each lane marker l_i, we set labels for each lane marker by ordering the relative distance from an image center. For example, we set the host left lane marker in TuSimple to label 0, the host right lane marker to label 1, and the remaining lane markers similarly to cover all N lane markers. For optimization, we used AdamW [20] with gradual warmup and cosine annealing learning rate schedule with initial learning rate as $8e^{-4}$. The weights λ_1 and λ_2 for loss function in Eq. 1 were set as 10 and 1, respectively. The number of shared HRM was fixed to 3 for all experiments and the number of channel C was set to 96. Each mini-batch has 14 images per GPU and we trained using 8 GPUs for 80 epochs on CULane and 140 epochs on Tusimple. Since we only recover the spatial resolution as the half size of an image, we resampled the result vertices to meet the original scale. To reduce overfitting, we applied Dropout with 0.1 probability after every HRM. Furthermore, we also applied data augmentation like random cropping, horizontal flipping, and photometric augmentations. In testing time, we set T_{vc}, *i.e.*, the threshold of vertex-wise existence confidence, as 0.6 and T_{lc}, *i.e.*, the threshold of lane marker-wise existence, as 0.5 for every experiment.

4.1. Results

Quantitative analysis: To verify the effectiveness of our method, we performed extensive comparisons with several state-of-the-art methods. Following [13], we evaluated mul-

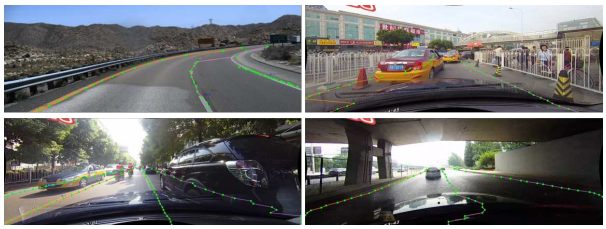


Figure 6. Failed examples from the CULane and TuSimple test sets.

Table 1. Comparison of different algorithms on the TuSimple test set.

Algorithm	Accuracy	FP	FN
ResNet-18 [13]	92.69%	0.0948	0.0822
ResNet-34 [13]	92.84%	0.0918	0.0796
LaneNet [22]	96.38%	0.0780	0.0244
EL-GAN [10]	96.39%	0.0412	0.0336
FCN-Instance [14]	96.5%	0.0851	0.0269
SCNN [25]	**96.53** %	0.0617	**0.0180**
R-18-SAD [13]	96.02%	0.0786	0.0451
R-34-SAD [13]	96.24%	0.0712	0.0344
R-18-E2E	96.04%	0.0311	0.0409
R-34-E2E	96.22 %	**0.0308**	0.0376
R-50-E2E	96.11 %	0.0321	0.0404
ERF-E2E	96.02 %	0.0321	0.0428

tiple backbones, *i.e.*, ResNet-18 (R-18-E2E), ResNet-34 (R-34-E2E), ResNet-50 (R-50-E2E), ERF (ERF-E2E) [29]. As illustrated in Table 1, the proposed method attained the competitive performance in the TuSimple dataset. Notable difference compared to other results is low FP ratio, which is obtained without complex post-processing like RANSAC. Interestingly, a heavier network happens to show lower accuracy numbers, *e.g.*, R-34-E2E versus R-50-E2E. The reason would be that the number of the TuSimple training images is not much enough to avoid the overfitting of the network.

In Table 3, the proposed method consistently outperforms the state-of-the-art methods in various scenarios of CULane dataset. Especially, the proposed method attained a better performance when comparing [19], which utilizes CycleGAN [35] to augment insufficient scenario data.

Qualitative analysis: Fig. 7 shows the localization of lane markers is successful at night, in the shadows, and when passing under the tunnel. Fig. 6 shows a few failure cases. The proposed method often fails when there exists reflection over the bonnet that makes it try to find a lane marker and when there are severe curves or occlusions.

4.2. Ablation Experiments

We investigated the effects of different choices of our proposed method, *e.g.*, the SE block existence and position, number of shared HRM layers and loss functions.

Table 2. Ablation study on different settings

ERFNet-E2E Architecture	CULane		
	Prec.	Recall	F-measure
Without SE	75.8	71.1	73.4
Pre-SE	75.7	71.5	73.5
Standard-SE	75.0	71.6	73.3
Post-SE	**76.5**	**71.8**	**74.0**

(a) **SE Position:** Results on the CULane dataset by changing the position of SE block in HRM.

R-18-E2E # shared	Flops ratio	TuSimple		
		Accuracy	FP	FN
1	1.00	**96.06**%	0.0316	0.0436
2	0.56	96.05%	0.0325	0.0419
3	0.34	96.04%	**0.0311**	**0.0410**
4	0.23	95.99%	0.0337	0.0443

(b) **Number of sharing pooling layers:** Results on the TuSimple dataset by changing the number of shared HRMs.

R-18-E2E Loss function	TuSimple		
	Accuracy	FP	FN
KL-divergence (KL)	95.49%	0.0376	0.0551
PL-Loss (PL)	95.69%	0.0455	0.0482
Cross-Entropy (CE)	**96.04**%	**0.0311**	**0.0410**

(c) **Loss function:** Results on the TuSimple dataset by changing the loss function.

Architecture: First, to confirm the pros of including SE block, we evaluated the effect of SE block position and existence on HRM layer in Table 2(a). Following the experiments in the original SE paper [15], we consider three variants: (1) Pre-SE block, in which the SE block is moved before the horizontal pixel unshuffle layer (see Fig. 2); (2) Standard-SE block, in which the SE block is after the residual operation, *i.e.*, after ConvBN in the residual branch; (3) Post-SE block, in which the SE block is moved after the summation of identity connection. Interestingly, in contrast to observations in the original SE paper [15], Post-SE performs much better than other configurations. It seems that the SE block at the end of the residual branch helps to recover the distinctiveness of lane markers whose information could be lost when squeezing the channel in the residual ConvBN layer (see Fig. 4).

The number of shared HRM: As discussed in Section 3.1, the number of shared HRMs is an important factor for the speed-accuracy trade-off. We changed the number of shared HRMs from 0 to 4 (accordingly the number of lane marker-wise HRMs varies from 6 to 2), and the results are summarized in Table 2(b). Note that the batch size of 8 is used in this experiment since the large number of shared HRMs requires much memory. As shown in Table 2(b), we can tune the number of shared HRM according to the speed-accuracy trade-off.

Loss function: To compare loss functions in terms of effectiveness, accuracy numbers per loss function are sum-

The results of

Figure 7. **E2E-LMD** using ERFNet as a backbone network on the CULane and TuSimple test images. All rows except the last one show the CULane test images. Green dots are appropriately sampled for visualization purpose. Best viewed in color.

Table 3. Comparison of different algorithms on the CULane test set. F_1-measure is displayed except "Cross" for which only FP is shown.

Category	**R-18-E2E**	**R-34-E2E**	**R-101-E2E**	**ERFNet-E2E**	R-18-SAD [13]	R-34-SAD [13]	R-101-SAD [13]	SCNN [25]	ERFNet[19]
Normal	90.0	90.4	90.1	91.0	89.8	89.9	90.7	90.6	**91.5**
Crowd	69.7	69.9	71.2	**73.1**	68.1	68.5	70	69.7	71.6
Highlight	60.2	61.5	60.9	64.5	59.8	59.9	59.9	58.5	**66**
Shadow	62.5	68.1	68.1	**74.1**	67.5	67.7	67	66.9	71.3
Arrow	83.2	83.7	84.3	85.8	83.9	83.8	84.4	84.1	**87.2**
Curve	70.3	69.8	70.2	**71.9**	65.5	66	65.7	64.4	71.6
Cross	2296	2077	2333	2022	1995	**1960**	2052	1990	2199
Night	63.3	63.2	65.2	**67.9**	64.2	64.6	66.3	66.1	67.1
No line	43.2	45.0	44.9	**46.6**	42.5	42.2	43.5	43.4	45.1
Total	70.8	71.5	71.9	**74.0**	70.5	70.7	71.8	71.6	73.1

marized in Table 2(c). Surprisingly, in our experiments, simple CE loss is preferable than others. The reason would be that the proposed horizontal reduction module helps to effectively incorporate spatial information between the ground truth position and the proximity between neighbors into a network, which leads to helping general CE loss to outperform other specially designed loss functions, *i.e.*, KL and PL losses.

5. Conclusion

In this paper, we proposed a new lane marker detection method to classify each lane marker and obtain its vertex in an end-to-end manner. A novel module for effective horizontal reduction has been devised, and with the module, the state-of-the-art performance is achieved without any complex post-processing. Although we designed the proposed architecture for the lane marker detection problem, it can be also used for other tasks, such as general polygon prediction and semantic/instance segmentation.In order to improve the proposed architecture in a better way, we plan to search the reduction module in an automatic manner.

References

[1] https://www.cnbc.com/2020/01/28/ubers-self-driving-cars-are-a-key-to-its-path-to-profitability.html. 1

[2] https://www.cnbc.com/2019/11/05/lyft-is-developing-self-driving-cars-at-its-level-5-lab-in-palo-alto.html. 1

[3] M. Aly. Real time detection of lane markers in urban streets. In *IEEE Intelligent Vehicles Symposium*, June 2008. 2

[4] Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Inf. Retr.*, 13(3):216–235, June 2010. 4

[5] Shriyash Chougule, Nora Koznek, Asad Ismail, Ganesh Adam, Vikram Narayan, and Matthias Schulze. Reliable multilane detection and classification by utilizing CNN as a regression network: Munich. In *ECCV Workshop*, 2018. 1, 2

[6] Murat Dikmen and Catherine M Burns. Autonomous driving in the real world: Experiences with tesla autopilot and summon. In *Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications*, pages 225–228, 2016. 1

[7] Wouter Van Gansbeke, Bert De Brabandere, Davy Neven, Marc Proesmans, and Luc Van Gool. End-to-end lane detection through differentiable least-squares fitting. In *ICCV Workshop*, 2019. 1, 2

[8] N. Garnett, S. Silberstein, S. Oron, E. Fetaya, U. Verner, A. Ayash, V. Goldner, R. Cohen, K. Horn, and D. Levi. Real-time category-based and general obstacle detection for autonomous driving. In *ICCV Workshop*, 2017. 1, 2

[9] Noa Garnett, Shai Silberstein, Shaul Oron, Ethan Fetaya, Uri Verner, Ariel Ayash, Vlad Goldner, Rafi Cohen, Kobi Horn, and Dan Levi. Real-time category-based and general obstacle detection for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 198–205, 2017. 1

[10] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. EL-GAN: Embedding loss driven generative adversarial networks for lane detection. In *ECCV Workshop*, 2019. 1, 2, 6

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015. 3

[12] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine vision and applications*, 25(3):727–745, 2014. 1

[13] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection CNNs by self attention distillation. In *ICCV*, 2019. 1, 2, 5, 6, 8

[14] Yen-Chang Hsu, Zheng Xu, Zsolt Kira, and Jiawei Huang. Learning to cluster for proposal-free instance segmentation. In *IJCNN*, 2018. 1, 2, 6

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 4, 6

[16] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011. 1

[17] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *ICCV*, 2017. 1, 2

[18] Dan Levi, Noa Garnett, and Ethan Fetaya. Stixelnet: A deep convolutional network for obstacle detection and road segmentation. In *BMVC*, 2015. 1, 2, 4

[19] Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, and Haowei Li. Lane detection in low-light conditions using an efficient data enhancement : Light conditions style transfer. *arXiv:2002.01177*, 2020. 1, 2, 6, 8

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[21] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*, 2016. 1

[22] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. *IEEE Intelligent Vehicles Symposium*, Jun 2018. 1, 2, 6

[23] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018. 1

[24] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W. Koh, Quoc V. Le, and Andrew Y. Ng. Tiled convolutional neural networks. In *NIPS*, 2010. 3

[25] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial CNN for traffic scene understanding. In *AAAI*, 2017. 1, 2, 3, 4, 5, 6, 8

[26] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv:1606.02147*, 2016. 1

[27] Sharon L Poczter and Luka M Jankovic. The google car: driving toward a better future? *Journal of Business Case Studies (JBCS)*, 10(1):7–14, 2014. 1

[28] Joseph P Robinson, Yuncheng Li, Ning Zhang, Yun Fu, and Sergey Tulyakov. Laplace landmark localization. In *ICCV*, 2019. 4

[29] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo. ERFNet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, Jan 2018. 1, 6

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[31] Daniel L Rosenband. Inside waymo's self-driving car: My favorite transistors. In *2017 Symposium on VLSI Circuits*, pages C20–C22. IEEE, 2017. 1

[32] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 3

[33] TuSimple. http://benchmark.tusimple.ai/#/t/1. 5

[34] David B Yoffie. Mobileye: The future of driverless cars. 2014. 1

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv:1703.10593*, 2017. 6