# CondLaneNet: a Top-to-down Lane Detection Framework Based on Conditional Convolution

Lizhe Liu[1] Xiaohao Chen[1] Siyu Zhu[1] Ping Tan[12]
[1]Alibaba Group [2]Simon Fraser University

## Abstract

*Modern deep-learning-based lane detection methods are successful in most scenarios but struggling for lane lines with complex topologies. In this work, we propose CondLaneNet, a novel top-to-down lane detection framework that detects the lane instances first and then dynamically predicts the line shape for each instance. Aiming to resolve lane instance-level discrimination problem, we introduce a conditional lane detection strategy based on conditional convolution and row-wise formulation. Further, we design the Recurrent Instance Module(RIM) to overcome the problem of detecting lane lines with complex topologies such as dense lines and fork lines. Benefit from the end-to-end pipeline which requires little post-process, our method has real-time efficiency. We extensively evaluate our method on three benchmarks of lane detection. Results show that our method achieves state-of-the-art performance on all three benchmark datasets. Moreover, our method has the coexistence of accuracy and efficiency, e.g. a 78.14 F1 score and 220 FPS on CULane. Our code is available at* `https://github.com/aliyun/conditional-lane-detection`.

## 1. Introduction

Artificial intelligence technology is increasingly being used in the driving field, which is conducive to autonomous driving and the Advanced Driver Assistance System(ADAS). As a basic problem in autonomous driving, lane detection plays a vital role in applications such as vehicle real-time positioning, driving route planning, lane-keeping assist, and adaptive cruise control.

Traditional lane detection methods usually rely on hand-crafted operators to extract features [24, 43, 13, 17, 15, 1, 16, 33], and then fit the line shape through post-processing such as Hough transform [24, 43] and Random Sampling Consensus (RANSAC) [17, 15]. However, traditional methods faile in maintaining robustness in real scene since the hand-crafted models cannot cope with the diversity of lane lines in different scenarios [27]. Recently, most studies about lane detection have focused on deep learning [34]. Early deep-learning-based methods detect lane lines through segmentation [28, 27]. Recently, various methods such as anchor-based methods [32, 2, 39], row-wise detection methods [30, 29, 41], and parametric prediction methods [31, 25] have been proposed and continue to refresh the accuracy and efficiency. Although deep-learning-based lane detection methods have made great progress [42], there are still many challenges.

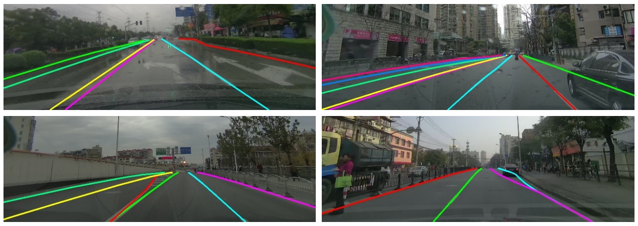


Figure 1. Scenes of lane lines with complex topologies. It is challenging to cope with the scenes such as the dense lines(the first row) and the fork lines(the second row). Different instances are represented by different colors in this figure.

A common problem for lane detection is instance-level discrimination. Most lane detection methods [27, 28, 19, 32, 12, 21, 29, 2, 30, 41, 39] predict lane points first and then aggregate the points into lines. But assigning different points to different lane instances [34] is still a common challenge. A simple solution is to label the lane lines into classes of a fixed number (e.g. labeled as 0, 1, 2, 3 if the maximum lane number is 4) and make a multi-class classification [28, 30, 41, 3]. But the limitation is that only a predefined, fixed number of lanes can be detected [27]. To overcome this limitation, the post-clustering strategy is investigated [27, 19]. However, this strategy is struggling for some cases such as dense lines. Another approach is anchor-based methods [25, 22, 39]. But it is not flexible to predict the line shape due to the fixed shape of the anchor.

Another challenge is the detection of lane lines with complex topologies, such as fork lines which have the same starting point, and dense lines which are parallel with close starting points. Such cases are common in driving scenarios, e.g. fork lines usually appear when the number of lanes changes. Homayounfar et al.[10] proposed an offline lane detection method for HDMap (High Definition Map) which can deal with the fork lines. However, there are few studies on the perception of lane lines with complex topologies for real-time driving scenarios.

The lane detection task is similar to instance segmentation, which requires assigning different pixels to different instances. Recently, some studies [35, 38] have investigated the conditional instance segmentation strategy, which is also promising for lane detection tasks. However, it is inefficient to directly apply this strategy to lane detection, since the constraint for the mask is not completely consistent with specifying the line shape [3, 34, 30].

In this work, we propose a novel lane detection framework called CondLaneNet. Aiming to resolve the lane instance-level discrimination problem, we propose the conditional lane detection strategy inspired by CondInst [35] and SOLOv2 [38]. Different from the instance segmentation tasks, we focus the optimization on specifying the lane line shape based on the row-wise formulation [30, 41]. Moreover, we design the Recurrent Instance Module(RIM) to deal with the detection of lane lines with complex topologies such as the dense lines and fork lines. Besides, benefit from the end-to-end pipeline that requires little post-process, our method achieves real-time efficiency. The contributions of this work are summarised as follows:

- We have greatly improved the ability of lane instance-level discrimination by the proposed conditional lane detection strategy and row-wise formulation.

- We solve the problem of detecting lane lines with complex topologies such as dense lines and fork lines by the proposed RIM.

- Our CondLaneNet framework achieves state-of-the-art performance on multiple datasets, e.g. an 86.10 F1 score (4.6% higher than SOTA) on CurveLanes and a 79.48 F1 score (3.2% higher than SOTA) on CULane. Moreover, the small version of our CondLaneNet has high efficiency while ensuring high accuracy, e.g. a 78.14 F1 score and 220 FPS on CULane.

## 2. Related Work

This section introduces the recent deep-learning-based lane detection methods. According to the strategy of line shape description, current methods can be divided into four categories: segmentation-based methods, anchor-based methods, row-wise detection methods, and parametric prediction methods.

### 2.1. Segmentation-based Methods

Segmentation-based methods [28, 12, 27, 19, 21, 6] are most common and have achieved impressive performance. Early methods [28, 6] used a multi-class classification strategy for lane instance discrimination. As explained in the previous section, this strategy is inflexible. For higher instance accuracy, the post-clustering strategy [4] was widely applied [27, 19]. Considering that the segmentation-based methods generally predict a down-scaled mask, some methods [19] predict an offset map for refinement. Recently, some studies [3, 30] indicated that it is inefficient to describe the lane line as a mask because the emphasis of segmentation is obtaining accurate classification per pixel rather than specifying the line shape. To overcome this problem, anchor-based methods and row-wise detection methods were proposed.

### 2.2. Anchor-based Methods

Anchor-based methods [32, 2, 39] take a top-to-down pipeline and focus the optimization on the line shape by regressing the relative coordinates. The predefined anchors can reduce the impact of the no-visual-clue problem [32] and improve the ability of instance discrimination. Due to the slender shape of lane lines, the widely used box-anchor in object detection [7] cannot be used directly. PointLaneNet [2] and CurveLane [39] used vertical lines as anchors. LaneATT [32] designed anchors with a slender shape and achieves state-of-the-art performance on multiple datasets. However, the fixed anchor shape results in a low degree of freedom in describing the line shape [39].

### 2.3. Row-wise Detection Methods

Row-wise detection methods [30, 29, 41] make good use of the shape prior and predict the line location for each row. In the training phase, the constraint on the overall line shape is realized through the location constraint of each row. Based on the continuity and consistency of the predicted locations from row to row, shape constraints can be added to the model [29, 30]. Besides, in terms of efficiency, some recent row-wise detection methods[30, 41, 11] have achieved advantages. However, instance-level discrimination is still the main problem for row-wise formulation. As the widely used post-clustering module [4] in segmentation-based methods [27, 19] cannot be directly integrated into the row-wise formulation, row-wise detection methods still take the multi-class classification strategy for lane instance discrimination. Considering the impressive performance on accuracy and efficiency, we also adopt the row-wise formulation in our method.

### 2.4. Parametric Prediction Methods

Different from the above methods which predict points, parametric prediction methods directly output parametric lines expressed by curve equation. PolyLaneNet [31] firstly proposed to use a deep network to regress the lane curve equation. LSTR [25] introduced transformer [37] to lane

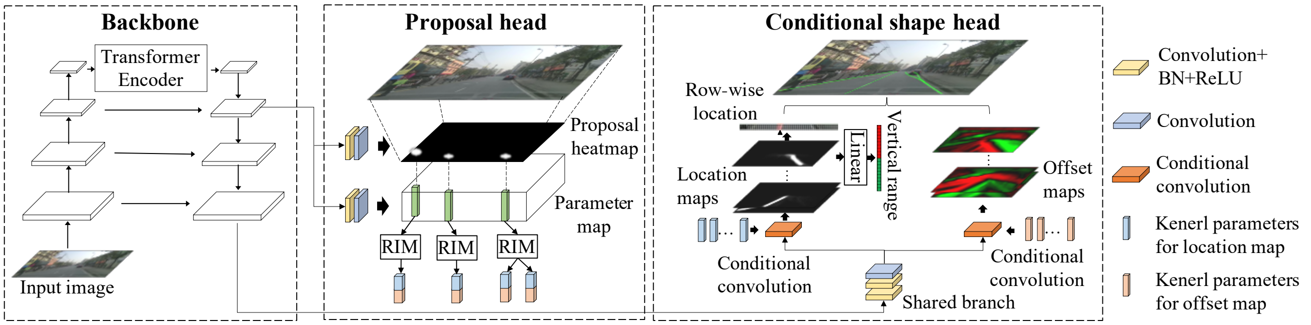


Figure 2. The structure of our CondLaneNet framework. The backbone adopts standard ResNet [8] and FPN [23] for multi-scale feature extraction. The transformer encoder module [37] is added for more efficient context feature extraction. The proposal head is responsible for detecting the proposal points which are located at the start point of the line. Meanwhile, a parameter map that contains the dynamic convolution kernels is predicted. The conditional shape head predicts the row-wise location, the vertical range, and the offset map to describe the shape for each line. To address the cases of dense lines and fork lines, the RIM is designed.

detection task and get 420fps detection speed. However, the parametric prediction methods have not surpassed other methods in terms of accuracy.

## 3. Methods

Given an input image $I \in R^{C\times H\times W}$, the goal of our CondLaneNet is to predict a collection of lanes $L = \{l_1, l_2, ..., l_N\}$, where $N$ is the total number of lanes. Generally, each lane $l_k$ is represented by an ordered set of coordinates as follows.

$$l_k = [(x_{k1}, y_{k1}), (x_{k2}, y_{k2}), ..., (x_{kN_k}, y_{kN_k})] \qquad (1)$$

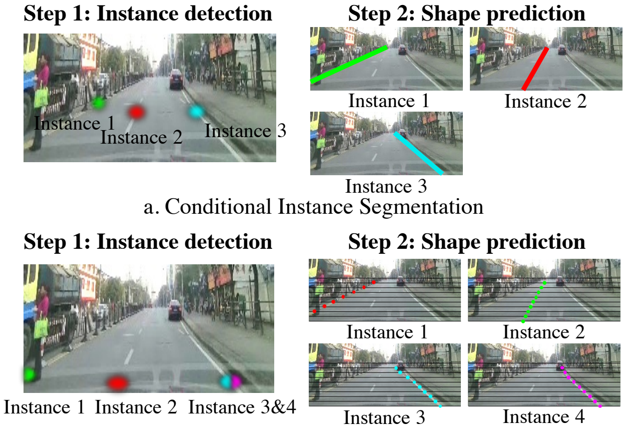
Where $k$ is the index of lane and $N_k$ is the max number of sample points of the $kth$ lane.

The overall structure of our CondLaneNet is shown in Figure 2. This section will first present the conditional lane detection strategy, then introduce the RIM(Recurrent Instance Module), and finally detail the framework design.

### 3.1. Conditional Lane Detection

Focusing on the instance-level discrimination ability, we propose the conditional lane detection strategy based on conditional convolution – a convolution operation with dynamic kernel parameters [14, 40]. The conditional detection process [35, 38] has two steps: instance detection and shape prediction, as is shown in Figure 3. The instance detection step predicts the object instance and regresses a set of dynamic kernel parameters for each instance. In the shape prediction step, conditional convolutions are applied to specify the instance shape. This process is conditioned on the dynamic kernel parameters. Since each instance corresponds to a set of dynamic kernel parameters, the shapes can be predicted instance-wisely.

This strategy has achieved impressive performance on instance segmentation tasks [35, 38]. However, directly applying the conditional instance segmentation strategy to lane detection is blunt and inappropriate. On the one hand, the segmentation-based shape description is inefficient for lane lines due to the excessively high degree of freedom [30]. On the other hand, the instance detection strategy for general objects is not suitable for slender and curved objects due to the inconspicuous visual characteristic of the border and the central. Our conditional lane detection strategy improves shape prediction and instance detection to address the above problems.



a. Conditional Instance Segmentation

b. Conditional Lane Detection

Figure 3. The difference between conditional instance segmentation and the proposed conditional lane detection strategy. Our CondLaneNet detects the start point of each line for instance detection and uses the row-wise formulation to describe the line shape instead of the mask. The overlapping lines can be distinguished based on the proposed RIM, which will be detailed in Section 3.2.

#### 3.1.1 Shape Prediction

We improve the row-wise formulation [30] to predict the line shape based on our conditional shape head, as is shown in Figure 2. In the row-wise formulation, we predict the lane location on each row and then aggregate the locations to get the lane line in the order from bottom to top, based on the prior of the line shape. Our row-wise formulation has three components: the row-wise location, the vertical range, and the offset map. The first two outputs are basic elements for most row-wise detection methods [30, 41]. Besides, we predict an offset map as the third output for further refinement.

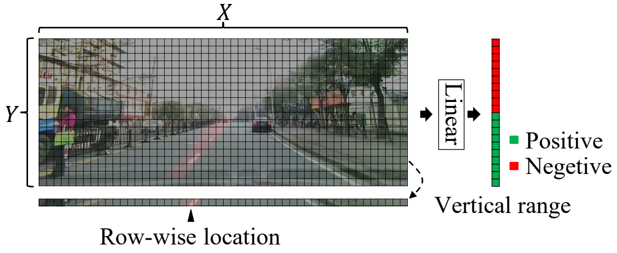


Figure 4. The process of parsing the row-wise location and the vertical range from the location map.

**Row-wise Location** As is shown in Figure 4, we divide the input image into grids of shape $Y \times X$ and predict a corresponded location map, which is a feature map of shape $1 \times Y \times X$ output by the proposed conditional shape head. On the location map, each row has an abscissa indicating the location of the lane line.

To get the row-wise location, a basic approach is to process the $X$-classes classification in each row. In inference time, the row-wise location is determined by picking the most responsive abscissa in each row. However, a common situation is that the line location is between the two grids, and both the two grids should have a high response. To overcome this problem, we introduce the following formulation.

For each row, we predict the probability that the lane line appears in each grid.

$$p_i = softmax(f_{loc}^i) \tag{2}$$

Where $i$ represents the $ith$ row, $f_{loc}^i$ is the feature vector of the $ith$ row of location map $f_{loc}$, $p_i$ is the probability vector for the $ith$ row.

The final row-wise location is defined as the expected abscissa.

$$E(\hat{x}_i) = \sum_j j \cdot p_{ij} \tag{3}$$

Where $E(\hat{x}_i)$ is the expected abscissa, $p_{ij}$ is the probability of the lane line passing through the coordinate $(j, i)$.

In the training phase, L1-loss is applied.

$$\ell_{row} = \frac{1}{N_v} \sum_{i \in V} |E(\hat{x}_i) - x_i| \tag{4}$$

Where $V$ represents the vertical range of the labeled line, $N_v$ is the number of valid rows.

**Vertical Range** The vertical lane range is determined by row-wisely predicting whether the lane line passes through the current row, as is shown in Figure 4. We add a linear layer and perform binary-classification row by row. We use the feature vector of each row in the location map as the input. The softmax-cross-entropy loss is adopted to guide the training process.

$$\ell_{range} = \sum_i (-y_{gt}^i log(v_i) - (1 - y_{gt}^i) log(1 - v_i)) \tag{5}$$

Where $v_i$ represents the predicted positive probability for $ith$ row and $y_{gt}^i$ is the groundtruth of $ith$ row.

**Offset Map** The row-wise location defined in Equation 3 points to the abscissa of the vertex on the left side of the grid, rather than the precise location. Thus, we add the offset map to predict the offset in the horizontal direction near the row-wise location for each row. We use L1-loss to constrain the offset map as follows.

$$\ell_{offset} = \frac{1}{N_\Omega} \sum_{(j,i) \in \Omega} \left| \hat{\delta}_{ij} - \delta_{ij} \right| \tag{6}$$

where $\hat{\delta}_{ij}$ and $\delta_{ij}$ are the predicted offset and the label offset on coordinate $(j, i)$. We define $\Omega$ as the area near the lane line with a fixed width. $N_\Omega$ is the number of pixels in $\Omega$.

**Shape Description** Each output lane line is represented as an ordered set of coordinates. For $kth$ line, the coordinate $(x_k^i, y_k^i)$ of the $ith$ row is represented as follows.

$$\begin{cases} y_k^i = H/Y \cdot i \\ x_k^i = W/X \cdot (loc_k^i + \delta(loc_k^i, i)) \end{cases} \tag{7}$$

Where $i \in \left[v_{min}^k, v_{max}^k\right]$, $v_{min}^k$ and $v_{max}^k$ are respectively the minimum and maximum values of the predicted vertical range, $loc_i^k$ is rounded down from $E_i^k$, $\delta(\cdot)$ is the predicted offset.

### 3.1.2 Instance Detection

We design the proposal head for instance detection, as is shown in Figure 2. For general conditional instance segmentation methods [35, 38], the instance is detected in an end-to-end pipeline by predicting the central of each object. However, it is hard to predict the central for the slender and curved lines because the visual characteristic of the line central is not obvious.

We detect the lane instance by detecting the proposal point located at the start point of the line. The start point has a more clear definition and more obvious visual characteristic than the central. We follow CenterNet [5] and predict a proposal heatmap to detect the proposal points and adopt focal loss following CornerNet [20] and CenterNet[5].

$$\ell_{point} = \frac{-1}{N_p}\sum_{xy}\begin{cases}(1-\hat{P}_{xy})^{\alpha}log(\hat{P}_{xy}) & P_{xy}=1\\(1-P_{xy})^{\beta}(\hat{P}_{xy})^{\alpha}log(1-\hat{P}_{xy}) & otherwise\end{cases} \tag{8}$$

Where $P_{xy}$ is the label at coordinate $(x,y)$ and $\hat{P}_{xy}$ is the predicted value at coordinate $(x,y)$ of the proposal heatmap. $N_p$ is the number of proposal points in the input image.

Besides, we regress the dynamic kernel parameters by predicting a parameter map following CondInst [35] and SOLOv2 [38]. The constraints of the parameter map are constructed through the constraints on the line shape.

### 3.2. Recurrent Instance Module

In the proposal head described above, each proposal point is bound to a lane instance. However, in practice, multiple lane lines can fall in the same proposal point such as the fork lines. To deal with the above cases, we propose the Recurrent Instance Module(RIM).

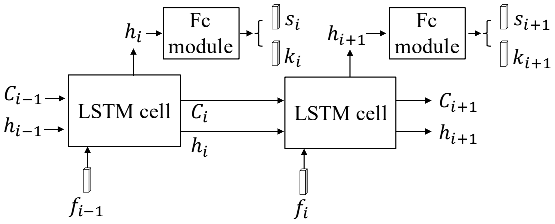


Figure 5. The Recurrent Instance Module. In this figure, $h$ and $c$ are the short-term memory and long-term memory respectively, $f$ is the input feature vector, $s$ is the output state logit, $k$ is the output kernel parameter vector.

The structure of the proposed RIM is shown in Figure 5. Based on LSTM(Long Short-term Memory) [9], the RIM recurrently predicts a state vector $s_i$ and a kernel parameter vector $k_i$, where $i$ is the index of the lines which start from the same proposal point. We define $s_i$ as two-dimensional logits that indicate two states: "continue" or "stop". The vector $k_i$ contains the kernel parameters for subsequent instance-wise dynamic convolution. In the inference phase, the RIM recurrently predicts the lane-wise kernel parameters bound to the same proposal point until the state is "stop". As is shown in Figure 2, RIM is added for each proposal point. Therefore, each proposal point can guide the shape prediction of multiple lane instances.

We adopt cross-entropy loss to constrain the state output as follows.

$$\ell_{state} = \frac{1}{N_s}\sum_{i} -[y_i \cdot log(s_i) + (1-y_i)\cdot log(1-s_i)] \tag{9}$$

Where $s_i$ is the output of softmax operation for $ith$ state, result $y_i$ is the ground truth for the $ith$ state and $N_s$ is the total number of the state outputs in a batch.

In the training phase, the total loss is defined as follows.

$$\ell_{total} = \ell_{point} + \alpha\ell_{row} + \beta\ell_{range} + \gamma\ell_{offset} + \eta\ell_{state} \tag{10}$$

The hyperparameters $\alpha$, $\beta$, $\gamma$ and $\eta$ are set to 1.0, 1.0, 0.4 and 1.0 respectively.

### 3.3. Architecture

The overall architecture is shown in Figure 2. We adopt ResNet [8] as the backbone and add a standard FPN [23] module to provide integrated multi-scale features. The proposal head detects the lane instances by predicting the proposal heatmap of shape $1 \times H_p \times W_p$. Meanwhile, a parameter map of shape $C_p \times H_p \times W_p$ that contains the dynamic kernel parameters is predicted. For the instance with the proposal point located at $(x_p, y_p)$, the corresponding dynamic kernel parameters are contained in the $C_p$ dimensional kernel feature vector at $(x_p, y_p)$ on the parameter map. Further, given the kernel feature vector, the RIM recurrently predicts the dynamic kernel parameters. Finally, the conditional shape head predicts the line shape instance-wisely conditioned on the dynamic kernel parameters.

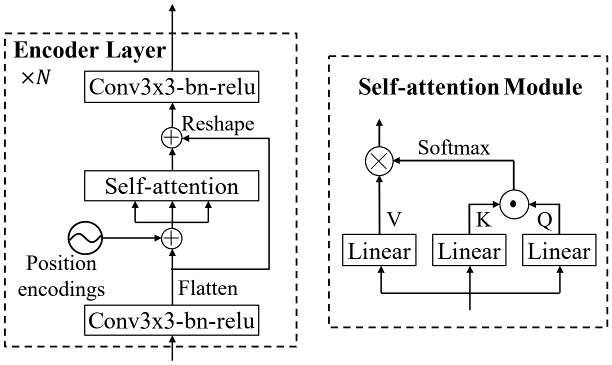


Figure 6. The structure of the transformer encoder. The ⊕, ⊙ and ⊗ respectively represent matrix addition, dot-product operation and element-wise product operation.

Our framework requires a strong capability of context feature fusion. For example, the prediction of the proposal point is based on the features of the entire lane line which generally has an elongated shape and long-range. Therefore, we add a transformer encoder structure to the last layer of the backbone for the fusion of contextual information. The structure of the transformer encoder used in our framework is shown in Figure 6.

# 4. Experiments

## 4.1. Experimental Setting

**Datasets** To extensively evaluate the proposed method, we conducte experiments on three benchmarks: CurveLanes [39], CULane [28], and TuSimple [36]. CurveLanes is a recently proposed benchmark with cases of complex topologies such as fork lines and dense lines. CULane is a widely used large lane detection dataset with 9 different scenarios. TuSimple is another widely used dataset of highway driving scenes. The details of the three datasets are shown in Tab. 1.

| Dataset | Train | Val. | Test | Road type | Fork |
|---|---|---|---|---|---|
| CurveLanes | 100K | 20K | 30K | Urban&Highway | √ |
| CULane | 88.9K | 9.7K | 34.7K | Urban&Highway | × |
| TuSimple | 3.3K | 0.4K | 2.8K | Highway | × |

Table 1. Details of three datasets.

**Evaluation Metrics** For CurveLanes and CULane, we adopte the evaluation metrics of SCNN [28] which utilizes the F1 measure as the metric. IoU between the predicted lane line and GT label is taken for judging whether a sample is true positive (TP) or false positive (FP) or false negative (FN). IoU of two lines is defined as the IoU of their masks with a fixed line width. Further, F1-measure is calculated as follows:

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

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

For TuSimple dataset [36], there are three official indicators: false-positive rate (FPR), false-negative rate (FNR), and accuracy.

$$accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}} \tag{14}$$

Where $C_{clip}$ is the number of correctly predicted lane points and $S_{clip}$ is the total number of lane points of a clip. Lane with accuracy greater than 85% is considered as a true-positive otherwise false positive or false negative. Besides, the F1 score is also reported.

**Implementation details** We fix the large, medium, and small versions of our CondLaneNet for all three datasets. The difference between the three models is shown in Table 2. For all three datasets, input images are resized to 800×320 pixels during training and testing. Since there are no cases of fork lines in CULane and TuSimple, RIM is only applied for the CurveLanes dataset. In the optimizing process, we use Adam optimizer [18] and step learning rate decay [26] with an initial learning rate of 3e-4. For each dataset, we train on the training set without any extra data. We respectively train 14, 16 and 70 epochs for CurveLanes, CULane and TuSimple with a batchsize of 32. The results are reported on the test set for CULane and TuSimple. For CurveLanes, we report the results on the validation set following CurveLane [39]. All the experiments were computed on a machine with an RTX2080 GPU.

| Model name | Backbone | Proposal head input | Shape head input |
|---|---|---|---|
| Large | Resnet-101 | downscale 16 | downscale 4 |
| Medium | Resnet-34 | downscale 16 | downscale 8 |
| Small | Resnet-18 | downscale 16 | downscale 8 |

Table 2. Difference of different versions of our CondLaneNet.

## 4.2. Results

The visualization results on the CurveLanes, CULane, and TuSimple datasets are shown in the Figure 7. The results show that our method can cope with complex line topologies. Even for the cases of dense lines and fork lines, our method can also successfully discriminate the instances.

| Method | F1 | Precision | Recall | FPS | GFlops(G) |
|---|---|---|---|---|---|
| SCNN [28] | 65.02 | 76.13 | 56.74 | | 328.4 |
| Enet-SAD [12] | 50.31 | 63.60 | 41.60 | | **3.9** |
| PointLaneNet [2] | 78.47 | 86.33 | 72.91 | | 14.8 |
| CurveLane-S [39] | 81.12 | **93.58** | 71.59 | | 7.4 |
| CurveLane-M [39] | 81.80 | 93.49 | 72.71 | | 11.6 |
| CurveLane-L [39] | 82.29 | 91.11 | 75.03 | | 20.7 |
| CondLaneNet-S | 85.09 | 87.75 | 82.58 | 154 | 10.3 |
| CondLaneNet-M | 85.92 | 88.29 | **83.68** | 109 | 19.7 |
| CondLaneNet-L | **86.10** | 88.98 | 83.41 | 48 | 44.9 |

Table 3. Comparison of different methods on CurveLanes.

**CurveLanes** The comparison results on CurveLanes are shown in Tabel 3. CurveLanes contains cases of lane lines with complex topologies such as curve, fork, and dense lanes. Our large version of CondLaneNet achieves a new state-of-the-art F1 score of 86.10, 4.63% higher than CurveLane-L. Our small version of CondLaneNet still has a performance of an 85.09 F1 score (3.40% higher than SOTA). Since our model can deal with cases of the fork and dense lane lines, there is a significant improvement in the recall indicator. Correspondingly, false-positive results will increase, resulting in a decrease in the precision indicator.

**CULane** The results of our CondLaneNet and other state-of-the-art methods on CULane are shown in Tabel 4. Our method achieves a new state-of-the-art result of a 79.48 F1 score, which has increased by 3.19%. Moreover, our method achieves the best performance in eight of nine scenarios, showing robustness to different scenarios. For some

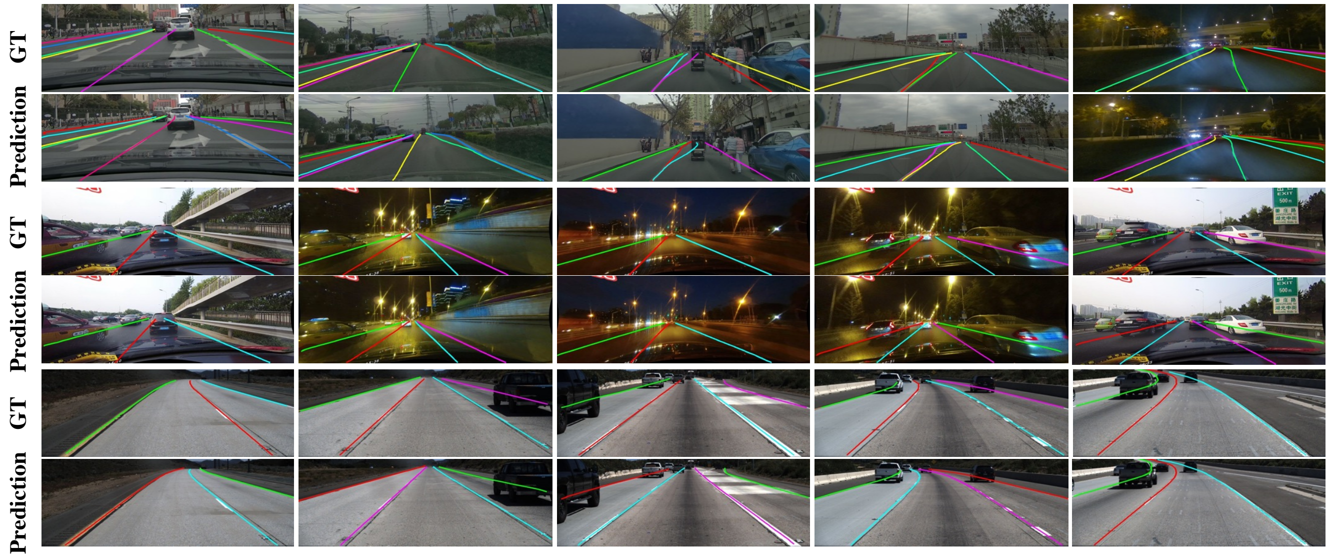


Figure 7. Visualization results on CurveLanes(the first row), CULane(the middle row) and TuSimple(the last row) datasets. Different lane instances are represented by different colors.

hard cases such as curve and night, our methods have obvious advantages. Besides, the small version of our CondLaneNet gets a 78.14 F1 score with a speed of 220 FPS, 1.12 higher and 8.5× speed than LaneATT-L. Compared with LaneATT-S, CondLaneNet-S achieves a 4.01 % F1 score improvement with similar efficiency. In most scenarios of CULane, the small version of our CondLaneNet exceeds all previous methods in the F1 measure.

**Tusimple** The results on TuSimple are shown in Table 5. Relatively, the gap between different methods on this dataset is smaller, due to the smaller amount of data and more single scenes. Our method achieves a new state-of-the-art F1 score of 97.24. Besides, the small version of our method gets a 97.01 F1 score with 220 FPS.

### 4.3. Ablation Study of Improvement Strategies

We performed ablation experiments on the CurveLanes dataset based on the small version of our CondLaneNet. The results are shown in Tabel 6. We take the lane detection model based on the original conditional instance segmentation strategy [35, 38] (as is shown in Figure 3.a) as the baseline. The first row shows the results of the baseline. In the second row, the proposed conditional lane detection strategy is applied and the lane mask expression is replaced by the row-wise formulation(as is shown in 3.b). In the third row, the offset map for post-refinement is added. In the fourth row, the transformer encoder is added and the offset map is removed. The fifth row presents the result of the model with the row-wise formulation, the offset map, and the transformer encoder. In the last row, RIM is added.

| Category | Total | Normal | Crowded | Dazzle | Shadow | No line | Arrow | Curve | Cross | Night | FPS | GFlops(G) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCNN [28] | 71.60 | 90.60 | 69.70 | 58.50 | 66.90 | 43.40 | 84.10 | 64.40 | 1990 | 66.10 | 7.5 | 328.4 |
| ERFNet-E2E [41] | 74.00 | 91.00 | 73.10 | 64.50 | 74.10 | 46.60 | 85.80 | 71.90 | 2022 | 67.90 | | |
| FastDraw [29] | | 85.90 | 63.60 | 57.00 | 69.90 | 40.60 | 79.40 | 65.20 | 7013 | 57.80 | 90.3 | |
| ENet-SAD [12] | 70.80 | 90.10 | 68.80 | 60.20 | 65.90 | 41.60 | 84.00 | 65.70 | 1998 | 66.00 | 75 | **3.9** |
| UFAST-ResNet34 [30] | 72.30 | 90.70 | 70.20 | 59.50 | 69.30 | 44.40 | 85.70 | 69.50 | 2037 | 66.70 | 175.0 | |
| UFAST-ResNet18 [30] | 68.40 | 87.70 | 66.00 | 58.40 | 62.80 | 40.20 | 81.00 | 57.90 | 1743 | 62.10 | **322.5** | |
| ERFNet-IntRA-KD [11] | 72.40 | | | | | | | | | | 100.0 | |
| CurveLanes-NAS-S [39] | 71.40 | 88.30 | 68.60 | 63.20 | 68.00 | 47.90 | 82.50 | 66.00 | 2817 | 66.20 | | 9.0 |
| CurveLanes-NAS-M [39] | 73.50 | 90.20 | 70.50 | 65.90 | 69.30 | 48.80 | 85.70 | 67.50 | 2359 | 68.20 | | 35.7 |
| CurveLanes-NAS-L [39] | 74.80 | 90.70 | 72.30 | 67.70 | 70.10 | 49.40 | 85.80 | 68.40 | 1746 | 68.90 | | 86.5 |
| LaneATT-Small [32] | 75.13 | 91.17 | 72.71 | 65.82 | 68.03 | 49.13 | 87.82 | 63.75 | **1020** | 68.58 | 250 | 9.3 |
| LaneATT-Medium [32] | 76.68 | 92.14 | 75.03 | 66.47 | 78.15 | 49.39 | 88.38 | 67.72 | 1330 | 70.72 | 171 | 18.0 |
| LaneATT-Large [32] | 77.02 | 91.74 | 76.16 | 69.47 | 76.31 | 50.46 | 86.29 | 64.05 | 1264 | 70.81 | 26 | 70.5 |
| CondLaneNet-Small | 78.14 | 92.87 | 75.79 | 70.72 | 80.01 | 52.39 | 89.37 | 72.40 | 1364 | 73.23 | 220 | 10.2 |
| CondLaneNet-Medium | 78.74 | 93.38 | 77.14 | **71.17** | 79.93 | 51.85 | 89.89 | 73.88 | 1387 | 73.92 | 152 | 19.6 |
| CondLaneNet-Large | **79.48** | **93.47** | **77.44** | 70.93 | **80.91** | **54.13** | **90.16** | **75.21** | 1201 | **74.80** | 58 | 44.8 |

Table 4. Comparison of different methods on CULane.

| Method | F1 | Acc | FP | FN | FPS | GFLOPS |
|---|---|---|---|---|---|---|
| SCNN [28] | 95.97 | 96.53 | 6.17 | **1.80** | 7.5 | |
| EL-GAN [6] | 96.26 | 94.90 | 4.12 | 3.36 | 10.0 | |
| PINet [19] | 97.21 | 96.70 | 2.94 | 2.63 | | |
| LineCNN [22] | 96.79 | **96.87** | 4.42 | 1.97 | 30.0 | |
| PointLaneNet [2] | 95.07 | 96.34 | 4.67 | 5.18 | 71.0 | |
| ENet-SAD [12] | 95.92 | 96.64 | 6.02 | 2.05 | 75.0 | |
| ERF-E2E [41] | 96.25 | 96.02 | 3.21 | 4.28 | | |
| FastDraw [29] | 93.92 | 95.20 | 7.60 | 4.50 | 90.3 | |
| UFAST-ResNet34 [30] | 88.02 | 95.86 | 18.91 | 3.75 | 169.5 | |
| UFAST-ResNet18 [30] | 87.87 | 95.82 | 19.05 | 3.92 | 312.5 | |
| PolyLaneNet [31] | 90.62 | 93.36 | 9.42 | 9.33 | 115.0 | 0.9 |
| LSTR [25] | 96.86 | 96.18 | 2.91 | 3.38 | **420** | **0.3** |
| LaneATT-ResNet18 [32] | 96.71 | 95.57 | 3.56 | 3.01 | 250 | 9.3 |
| LaneATT-ResNet34 [32] | 96.77 | 95.63 | 3.53 | 2.92 | 171 | 18.0 |
| LaneATT-ResNet122 [32] | 96.06 | 96.10 | 5.64 | 2.17 | 26 | 70.5 |
| CondLaneNet-S | 97.01 | 95.48 | 2.18 | 3.80 | 220 | 10.2 |
| CondLaneNet-M | 96.98 | 95.37 | 2.20 | 3.82 | 154 | 19.6 |
| CondLaneNet-L | **97.24** | 96.54 | **2.01** | 3.50 | 58 | 44.8 |

Table 5. Comparison of different methods on TuSimple.

| Baseline | Row-wise | Offset | Encoder | RIM | F1 score |
|---|---|---|---|---|---|
| √ | | | | | 72.19 |
| | √ | | | | 80.09(+7.9) |
| | √ | √ | | | 81.24(+9.05) |
| | √ | | √ | | 81.85(+9.66) |
| | √ | √ | √ | | 83.41(+11.22) |
| | √ | √ | √ | √ | 85.09(+12.90) |

Table 6. Ablation study of the improvement strategies on CurveLanes base on the small version of our CondLaneNet.

Comparing the first two rows, we can see that the proposed conditional lane detection strategy has significantly improved the performance. Comparing the results of the 2nd and the 3rd row, the 4th and the 5th row, we can see the positive effect of the offset map. Moreover, the transformer encoder plays a vital role in our framework, which can be indicated by comparing the 2nd and the 4th row, the 3rd and the 5th row. Besides, the proposed RIM designed for the fork lines and dense lines also improves the accuracy. As is shown in Figure 8, the lanes lines with complex topologies including fork lines and dense lines can be detected by CondLaneNet with RIM.

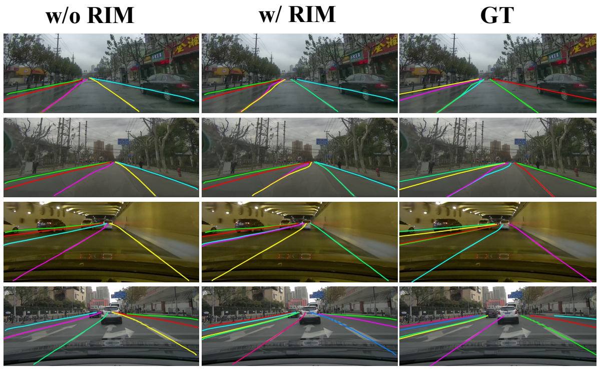


Figure 8. Results of CondLaneNet with and without RIM.

## 4.4. Ablation Study of Transformer Encoder

This section further analyzes the function of the transformer encoder which indicates a vital role in the previous experiments. Our method first detects instances by detecting the proposal points and then predicts the shape for each instance. The accuracy of the proposal points greatly affects the final accuracy of the lane lines. We design different control groups to compare the accuracy of the proposal points and lane lines on CurveLanes. We define the proposal points which locate in the eight neighborhoods of the groundtruth points as the true-positive samples. We report the F1 score of the proposal points and lane lines, as is shown in Table 7.

| Model | Small | | Medium | | Large | |
|---|---|---|---|---|---|---|
| Target | P. point | Line | P. point | Line | P. point | Line |
| Standard | 88.35 | 85.09 | 88.99 | 85.92 | 89.54 | 86.10 |
| S. w/o encoder | 85.51 | 82.97 | 88.68 | 85.91 | 89.33 | 85.98 |
| Hacked | 88.05 | 84.39 | 88.90 | 85.93 | 89.37 | 85.99 |

Table 7. Ablation study of the transformer encoder module.

The first row shows the results of the small, medium and large versions of the standard CondLaneNet. In the second row, the transformer encoder is removed. In the third row, we hack the inference process of the second row by replacing the proposal heatmap with the proposal heatmap output by the standard model(the first row). For the small version, removing the encoder leads to a significant drop for both proposal points and lanes. However, using the proposal heatmap of the standard model, the results on the third row are close to the first row.

The above results prove that the function of the encoder is mainly to improve the detection of the proposal points, which rely on contextual features and global information. Besides, the contextual features can be more fully refined in deeper networks. Therefore, for the medium and large versions, the improvement of the encoder is far less than the small version.

## 5. Conclusion

In this work, We proposed CondLaneNet, a novel top-to-down lane detection framework that detects the lane instances first and then instance-wisely predict the shapes. Aiming to resolve the instance-level discrimination problem, we proposed the conditional lane detection strategy based on conditional convolution and row-wise formulation. Moreover, we designed RIM to cope with complex lane line topologies such as dense lines and fork lines. Our CondLaneNet framework refreshed the state-of-the-art performance on CULane, CurveLanes, and TuSimple. Moreover, on CULane and CurveLanes, the small version of our CondLaneNet not only surpassed other methods in accuracy, but also presented real-time efficiency.

## References

[1] Amol Borkar, Monson Hayes, and Mark T Smith. Robust lane detection and tracking with ransac and kalman filter. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3261–3264, 2009.

[2] Zhenpeng Chen, Qianfei Liu, and Chenfan Lian. Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 2563–2568, 2019.

[3] 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. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[4] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[5] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019.

[6] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2911–2920, 2019.

[11] Yuenan Hou, Zheng Ma, Chunxiao Liu, Tak-Wai Hui, and Chen Change Loy. Inter-region affinity distillation for road marking segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12486–12495, 2020.

[12] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1013–1021, 2019.

[13] Junhwa Hur, Seung-Nam Kang, and Seung-Woo Seo. Multilane detection in urban driving environments using conditional random fields. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1297–1302, 2013.

[14] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, page 667–675, 2016.

[15] Ruyi Jiang, Reinhard Klette, Tobi Vaudrey, and Shigang Wang. New lane model and distance transform for lane detection and tracking. In *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, pages 1044–1052, 2009.

[16] Yan Jiang, Feng Gao, and Guoyan Xu. Computer vision-based multiple-lane detection on straight road and in a curve. In *Proceedings of the International Conference on Image Analysis and Signal Processing*, pages 114–117, 2010.

[17] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26, 2008.

[18] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[19] Yeongmin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection. *arXiv preprint arXiv:2002.06604*, 2020.

[20] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[21] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 1947–1955, 2017.

[22] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. Line-cnn: End-to-end traffic line detection with line proposal unit. *IEEE Transactions on Intelligent Transportation Systems*, 21(1):248–258, 2019.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[24] Guoliang Liu, Florentin Wörgötter, and Irene Markelić. Combining statistical hough transform and particle filter for robust lane detection and tracking. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 993–997, 2010.

[25] Ruijin Liu, Zejian Yuan, Tie Liu, and Zhiliang Xiong. End-to-end lane shape prediction with transformers. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 3694–3702, 2021.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

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

[28] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[29] Jonah Philion. Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11582–11591, 2019.

[30] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 276–291, 2020.

[31] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Polylanenet: Lane estimation via deep polynomial regression. In *Proceedings of the International Conference on Pattern Recognition*, 2020.

[32] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixão, Claudine Badue, Alberto F De Souza, and Thiago Olivera-Santos. Keep your eyes on the lane: Attention-guided lane detection. *arXiv preprint arXiv:2010.12035*, 2020.

[33] Huachun Tan, Yang Zhou, Yong Zhu, Danya Yao, and Keqiang Li. A novel curve lane detection based on improved river flow and ransa. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems*, pages 133–138, 2014.

[34] Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 2020.

[35] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[36] TuSimple. Tusimple lane detection benchmark, 2017. https://github.com/TuSimple/tusimple-benchmark.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[38] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. In *Advances in Neural Information Processing Systems*, pages 17721–17732, 2020.

[39] Hang Xu, Shaoju Wang, Xinyue Cai, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704, 2020.

[40] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems*, 2019.

[41] Seungwoo Yoo, Hee Seok Lee, Heesoo Myeong, Sungrack Yun, Hyoungwoo Park, Janghoon Cho, and Duck Hoon Kim. End-to-end lane marker detection via row-wise classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1006–1007, 2020.

[42] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[43] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, and Huiyan Chen. A novel lane detection based on geometrical model and gabor filter. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 59–64, 2010.