CLRNet: Cross Layer Refinement Network for Lane Detection

Tu Zheng[1,2*]   Yifei Huang[1,2*]   Yang Liu[1,2]   Wenjian Tang[1]   Zheng Yang[1]
Deng Cai[1,2]   Xiaofei He[1,2]
[1]Fabu   [2]Zhejiang University

Abstract

Lane is critical in the vision navigation system of the intelligent vehicle. Naturally, lane is a traffic sign with high-level semantics, whereas it owns the specific local pattern which needs detailed low-level features to localize accurately. Using different feature levels is of great importance for accurate lane detection, but it is still under-explored. In this work, we present Cross Layer Refinement Network (CLRNet) aiming at fully utilizing both high-level and low-level features in lane detection. In particular, it first detects lanes with high-level semantic features then performs refinement based on low-level features. In this way, we can exploit more contextual information to detect lanes while leveraging local detailed lane features to improve localization accuracy. We present ROIGather to gather global context, which further enhances the feature representation of lanes. In addition to our novel network design, we introduce Line IoU loss which regresses the lane line as a whole unit to improve the localization accuracy. Experiments demonstrate that the proposed method greatly outperforms the state-of-the-art lane detection approaches.

1. Introduction

Lane detection is an important yet challenging task in computer vision, which requires the network to predict lanes in an image. Detecting lanes can benefit many applications, such as autonomous driving and the Advanced Driver Assistance System (ADAS), which helps intelligent vehicles localize themselves better and drive safer.

Benefiting from the effective feature representation of CNN, many approaches [17, 19, 33] have obtained promising performance. However, there are still some challenges for detecting accurate lanes. Lane has high-level semantics, whereas it owns the specific local pattern which needs detailed low-level features to localize accurately. How to utilize different feature levels effectively in CNN remains a problem. As we can see in Fig. 1(a), the landmark and
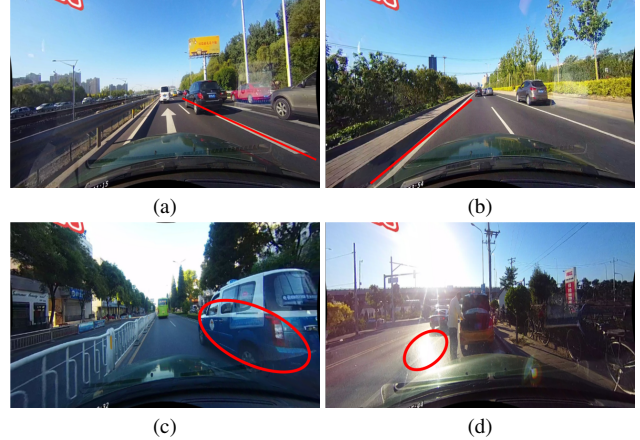
---
*Equal contribution.

Figure 1. Illustrations of hard cases for lane detection. (a) The detection result of low-level features. It mistakes landmark as lane due to losing global context. (b) The detection result of high-level features. It predicts inaccurate localization of the lane. (c) The case that lane is almost occupied by the car. (d) The case that lane is blurred by the extreme lighting condition.

lane line have different semantics, but they share the similar feature (*e.g.*, the long white line). It is hard to distinguish them without high-level semantics and global context. On the other hand, the locality is also essential since lane is long and thin with the simple local pattern. We show the detection result of high-level features in Fig 1(b), though the lane is detected, its location is not precise. Thus, the low-level and high-level information are complementary for accurate lane detection. Previous works either model local geometry of lanes and integrate them into global results [20] or construct a fully-connected layer with global features to predict lanes [19]. These detectors have demonstrated the importance of local or global features for lane detection, but they don't take advantage of both features, yielding inaccurate detection performance.

Another common problem in lane detection is no visual evidence for the presence of lanes. As shown in Fig. 1(c), the lane is occupied by the car while in Fig. 1(d), the lane is hard to recognize due to the extreme lighting condition. In the literature, SCNN [17] and RESA [33] propose a

message-passing mechanism to gather global context, but these methods perform pixel-wise prediction and don't take lane as a whole unit. Thus their performances lag behind many state-of-the-art detectors.

In this paper, we propose a new framework, Cross Layer Refinement Network (CLRNet), which fully utilizes low-level and high-level features for lane detection. Specifically, we first perform detection in high semantic features to coarsely localize lanes. Then, we perform refinement based on fine-detail features to get more precise locations. Progressively refining the location of lane and feature extraction leads to high accuracy detection results. To solve the problem of non-visual evidence of lane, we introduce ROIGather to capture more global contextual information by building the relation between the ROI lane feature and the whole feature map. Moreover, we define the IoU of lane lines and propose the Line IoU (LIoU) loss to regress the lane as a whole unit and considerably improve the performance compared with standard loss, *i.e.*, smooth-l_1 loss.

We demonstrate the effectiveness of our method on three lane detection benchmarks, *i.e.*, CULane [17], Tusimple [26], and LLAMAS [2]. The experiment results show our method achieves state-of-the-art accuracy on all datasets. The main contributions can be summarized as follows:

- We demonstrate low-level and high-level features are complementary for lane detection, and we propose a novel network architecture (CLRNet) to fully utilize low-level and high-level features for lane detection.

- We propose ROIGather to further enhance the representation of lane features by gathering global context, which can also be plugged into other networks.

- We propose Line IoU (LIoU) loss tailored for lane detection, regressing the lane as the whole unit and considerably improving the performance.

- To better compare the localization accuracy of different detectors, we also adopt the new mF1 metrics. We demonstrate the proposed method greatly outperforms other state-of-the-art approaches on three lane detection benchmarks.

2. Related Work

According to the representation of lane, current CNN-based lane detection can be divided into three categories: segmentation-based method, anchor-based method, and parameter-based method.

2.1. Segmentation-based methods

Modern algorithms typically adopt a pixel-wise prediction formulation, *i.e.*, treat lane detection as a semantic segmentation task. SCNN [17] proposes a message-passing mechanism to address no visual evidence problem, which captures the strong spatial relationship for lanes. SCNN significantly improves the lane detection performance, but the method is slow for real-time application. RESA [33] proposes a real-time feature aggregation module, enabling the network to gather the global feature and improve performance. In CurveLane-NAS [28], they use neural architecture search (NAS) to find a better network for capturing accurate information to benefit the detection of curve lanes. However, the NAS is extremely expensive computationally and costs huge GPU hours. These segmentation-based methods are ineffective and time-consuming since they perform pixel-wise prediction on the whole image and don't consider lanes as a whole unit.

2.2. Anchor-based methods

Anchor-based methods in lane detection can be divided into two classes, *e.g.*, line anchor-based methods and row anchor-based methods. Line anchor-based methods adopt predefined line anchors as references to regress accurate lanes. Line-CNN [8] is the pioneering work to use line anchors in lane detection. LaneATT [24] proposes a novel anchor-based attention mechanism that aggregates global information. It achieves state-of-the-art results and shows both high efficacy and efficiency. SGNet [22] introduces a novel vanish-point guided anchor generator and adds multiple structural guidance to improve performance. As for the row anchor-based method, it predicts the probable cell for each predefined row on images. UFLD [19] first proposes a row anchor-based lane detection method and adopts lightweight backbones to achieve high inference speed. Albeit simple and fast, its overall performance is not good. CondLaneNet [12] introduces a conditional lane detection strategy based on conditional convolution and row anchor-based formulation, *i.e.*, it first locates start points of lane lines then performs row anchor-based lane detection. However, start points are hard to recognize in some complex scenarios, which results in relatively inferior performance.

2.3. Parameter-based methods

Different from points regression, parameter-based methods model the lane curve with parameters and regress these parameters to detect lanes. PolyLaneNet [25] adopts a polynomial regression problem and achieves high efficiency. LSTR [13] takes road structures with camera pose into account to model the lane shape, then introduces the transformer to lane detection task to get the global feature. Parameter-based methods have fewer parameters to regress, but they are sensitive to the predicted parameters, *e.g.*, the error prediction on high-order coefficient may cause shape change of lanes. Though parameter-based methods have fast inference speed, they still struggle to achieve higher performance.
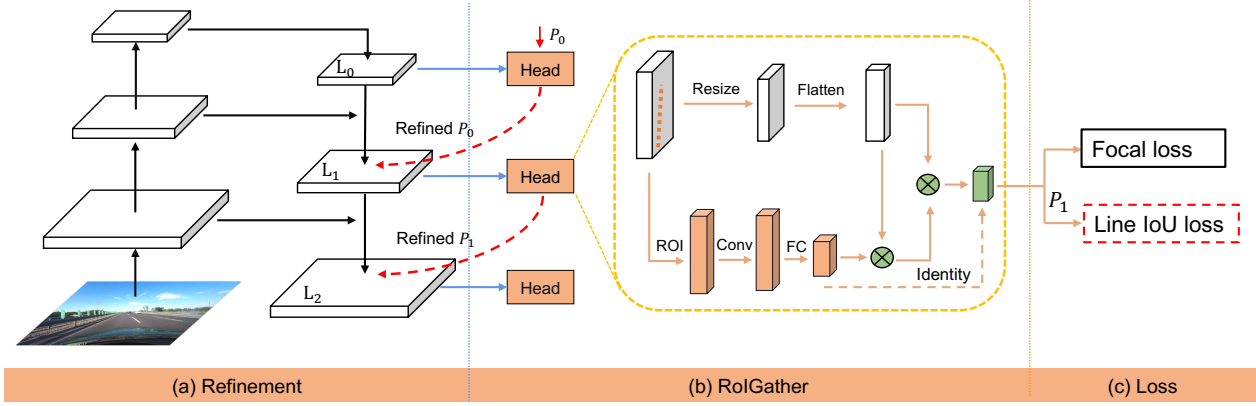
| (a) Refinement | (b) RoIGather | (c) Loss |

Figure 2. Overview of the proposed **CLRNet**. (a) The network generates feature maps from FPN [9] structure. Subsequently, each lane prior will be refined from high-level features to low-level features. (b) Each head will exploit more contextual information for lane prior features. (c) Classification and regression of lane priors. The proposed Line IoU loss helps further improve the regression performance.

3. Approach

3.1. The Lane Representation

Lane Prior. Lanes are thin and long with strong shape priors, thus a predefined lane prior can help the network better localize lanes. In common object detection, objects are represented by rectangular boxes. Nevertheless, the box is not appropriate for the representation of the long line. Following [8] and [24], we use equally-spaced 2D-points as lane representation. Specifically, lane is expressed as a sequence of points, *i.e.*, $P = \{(x_1, y_1), \cdots, (x_N, y_N)\}$. The y coordinate of points is equally sampled through image vertically, *i.e.*, $y_i = \frac{H}{N-1} * i$, where H is image height. Accordingly, the x coordinate is associated with the respective $y_i \in Y$. In our paper, we call this representation **Lane Prior**. Each lane prior will be predicted by the network and consists of four components: (1) foreground and background probabilities. (2) the length of lane prior. (3) the start point of the lane line and the angle between the x-axis of the lane prior (termed as x, y, and θ). (4) The N offsets, *i.e.*, the horizontal distance between the prediction and its ground truth.

3.2. Cross Layer Refinement

Motivation. In neural networks, deep high-level features strongly respond to entire objects with more semantic meanings, while the shallow low-level features are with more local contextual information. Allowing lane objects to access high-level features can help exploit more useful context information, *e.g.*, to distinguish lane lines or landmarks. In the meantime, fine-detail features help detect lanes with high localization accuracy. In object detection [9], it builds the feature pyramid to leverage the pyramidal shape of a ConvNet's feature hierarchy and assigns different scales of objects to different pyramid levels. However, it is hard to directly assign a lane to only one level since high-level and

low-level features are both critical for lanes. Inspired by Cascade RCNN [3], we can assign lane objects to all levels and detect lanes sequentially. In particular, we can detect lanes with high-level features to localize lanes coarsely. Based on the detected lanes, we can refine them with more detailed features.

Refinement structure. Our goal is to leverage a ConvNet's pyramidal feature hierarchy, which has semantics from low to high levels, and build a feature pyramid with high-level semantics throughout. We take ResNet [6] as the backbone and use $\{L_0, L_1, L_2\}$ to denote feature levels generated by FPN. As shown in Fig. 2, our cross layer refinement starts from the highest level L_0 and gradually approaches L_2. We use $\{R_0, R_1, R_2\}$ to denote the corresponding refinements. Then we can build a sequence of refinements

$$P_t = P_{t-1} \circ R_t(L_{t-1}, P_{t-1}), \tag{1}$$

where $t = 1, \cdots, T$, T is the total number of refinements. Our method performs detection from highest level layer with high semantics. P_t is the parameter of lane prior (start point coordinate x, y and angle θ), which is learnable inspired by [23]. For the first layer L_0, the P_0 is uniformly distributed on image plane. The refinement R_t takes the P_t as input to get the ROI lane features and then performs two FC layers to get the refined parameter P_t. Progressively refining the lane prior and feature extraction is important for the success of cross layer refinement. Note that, our method is not limited to FPN structure, only using ResNet [6] or adopting PAFPN [14] is also suitable.

3.3. ROIGather

Motivation. After we assign lane priors to each feature map, we can get features of lane priors with ROIAlign [5]. However, the contextual information of these features is still

not sufficient. In some cases, the lane instance may be occupied or blurred with extreme lighting conditions. Thus there is no local visual evidence for the presence of lane. To determine whether a pixel belongs to a lane, we need to look at nearby features. Some recent studies [27, 32] also indicate that the performance could be improved if making sufficient use of long-range dependencies. Thus, we can gather more useful contextual information to better learn lane feature. To this end, we add convolutions along the lane prior. In this way, each pixel in the lane prior can gather information of nearby pixels, and occupied parts can be reinforced from that information. Moreover, we build relations between features of lane priors and the whole feature map. Thus, it can exploit more contextual information to learn better feature representations.

ROIGather structure. The ROIGather module is light-weighted and easy to implement. It takes feature map and lane priors as input, each lane prior has N points. For each lane prior, we follow ROIAlign [5] to get the ROI feature of lane prior ($\mathcal{X}_p \in \mathbb{R}^{C \times N_p}$). Unlike ROIAlign for bounding box, we uniform sample N_p points from the lane prior and use bilinear interpolation to compute the exact values of input features at these locations. For ROI features of L_1, L_2, we concatenate the ROI features of previous layers to enhance feature representations. Convolutions are performed on the extracted ROI features to gather nearby features for each lane pixel. To save memory, we use fully-connected to further extract the lane prior feature ($\mathcal{X}_p \in \mathbb{R}^{C \times 1}$). The feature map is resized to $\mathcal{X}_f \in \mathbb{R}^{C \times H \times W}$ and flattened to $\mathcal{X}_f \in \mathbb{R}^{C \times HW}$. Detail settings are in Sec. 4.2.

To gather the global context for features of lane priors, we first compute the attention [27] matrix $\mathcal{W}$ between ROI lane prior feautre ($\mathcal{X}_p$) and the global feature map ($\mathcal{X}_f$), which is written as:

$$\mathcal{W} = f(\frac{\mathcal{X}_p^T \mathcal{X}_f}{\sqrt{C}}), \quad (2)$$

where f is a normalize function $softmax$. The aggregated feature is written as:

$$\mathcal{G} = \mathcal{W} \mathcal{X}_f^T. \quad (3)$$

The output $\mathcal{G}$ reflects the bonus of $\mathcal{X}_f$ to $\mathcal{X}_p$ which is selected from all locations of $\mathcal{X}_f$. Finally, we add the output to the original input X_p.

3.4. Line IoU loss

Motivation. As discussed above, the lane prior consists of discrete points needed to be regressed with its ground truth. The commonly used distance loss like smooth-l_1 can be used to regress these points. However, this kind of loss takes points as separate variables, which is an oversimplified assumption [31], resulting in less accurate regression.
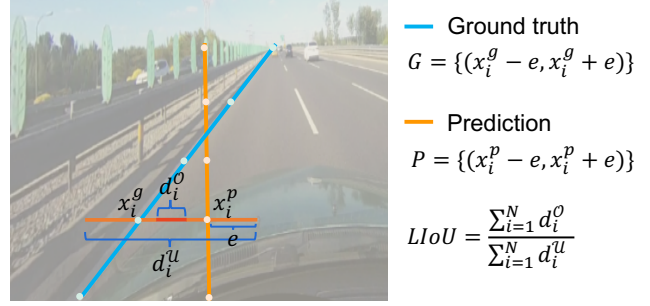

Figure 3. Illustration of Line IoU. Line IoU (interaction over union) can be calculated by integrating the IoU of the extended segment in terms of sampled x_i position.

In contrast to distance loss, Intersection over Union (IoU) can take the lane prior as a whole unit to regress and it is tailored for evaluation metric [21, 31, 34]. In our work, we derive an easy and effective algorithm to compute the Line IoU (LIoU) loss.

Formula. We introduce Line IoU loss starting from the definition of the line segment IoU, which is the ratio of interaction over union between two line segments. For each point in the predicted lane as shown in Fig. 3, we first extend it (x_i^p) with a radius e into a line segment. Then IoU can be calculated between the extended line segment and its ground truth, which is written as:

$$IoU = \frac{d_i^{\mathcal{O}}}{d_i^{\mathcal{U}}} = \frac{\min(x_i^p + e, x_i^g + e) - \max(x_i^p - e, x_i^g - e)}{\max(x_i^p + e, x_i^g + e) - \min(x_i^p - e, x_i^g - e)}, \quad (4)$$

where $x_i^p - e, x_i^p + e$ are the extended points of x_i^p, $x_i^g - e, x_i^g + e$ are the corresponding ground truth points. Note that, $d_i^{\mathcal{O}}$ can be negative, which can make it feasible to optimize in case of non-overlapping line segments.

Then LIoU can be considered as the combination of infinite line points. To simplify the expression and make it easy to compute, we transform it into a discrete form,

$$LIoU = \frac{\sum_{i=1}^{N} d_i^{\mathcal{O}}}{\sum_{i=1}^{N} d_i^{\mathcal{U}}}. \quad (5)$$

Then, the LIoU loss is defined as

$$\mathcal{L}_{LIoU} = 1 - LIoU, \quad (6)$$

where $-1 \leq LIoU \leq 1$, when two lines overlay perfectly, then $LIoU = 1$, $LIoU$ converges to -1 when two lines are far away.

Our Line IoU loss exhibits two advantages: (1) It is simple and differentiable, which is very easy to implement parallel computations. (2) It predicts the lane as a whole unit, which helps improve the overall performance.

4

3.5. Training and Inference Details

Positive samples selection. During training, each ground truth lane is assigned with one or more predicted lanes dynamically as positive samples, which is inspired by [4]. In particular, we first sort the predicted lanes based on the assigning cost, which is defined as:

$$\mathcal{C}_{assign} = w_{sim}\mathcal{C}_{sim} + w_{cls}\mathcal{C}_{cls}, \tag{7}$$
$$\mathcal{C}_{sim} = (\mathcal{C}_{dis} \cdot \mathcal{C}_{xy} \cdot \mathcal{C}_{theta})^2.$$

Here $\mathcal{C}_{cls}$ is the focal cost [10] between predictions and labels. $\mathcal{C}_{sim}$ is the similarity cost between predicted lanes and ground truth. It consists of three parts, $\mathcal{C}_{dis}$ means the average pixel distance of all valid lane points, $\mathcal{C}_{xy}$ means the distance of start point coordinates, $\mathcal{C}_{theta}$ means the difference of the theta angle, they are all normalized to $[0, 1]$. w_{cls} and w_{sim} are weight coefficients of each defined component. Each ground truth lane is assigned with a dynamic number (top-k) of predicted lanes based on $\mathcal{C}_{assign}$.

Training Loss. Training loss consists of classification loss and regression loss. The regression loss is only performed on the assigned samples. The overall loss function is defined as:

$$\mathcal{L}_{total} = w_{cls}\mathcal{L}_{cls} + w_{xytl}\mathcal{L}_{xytl} + w_{LIoU}\mathcal{L}_{LIoU}. \tag{8}$$

$\mathcal{L}_{cls}$ is the focal loss between predictions and labels, $\mathcal{L}_{xytl}$ is the smooth-l_1 loss for the start point coordinate, theta angle and lane length regression, $\mathcal{L}_{LIoU}$ is the Line IoU loss between the predicted lane and ground truth. Optionally, we can add an auxiliary segmentation loss following [19]. It is only used in the training period and has no cost in inference.

Inference. We set a threshold with a classification score to filter the background lanes (low score lane priors), and we use nms to remove high-overlapped lanes following [24]. Our method can also be nms-free if we use the one-to-one assignment, i.e., set the top-k = 1.

4. Experiment

4.1. Datasets

We conduct experiments on two widely used lane detection benchmark datasets: **CULane** [17] and **Tusimple** [26] and one recently released benchmark (**LLAMAS** [2]).

CULane [17] is a large scale challenging dataset for lane detection. It contains nine challenging categories, such as crowded, night, cross, etc. The CULane dataset consists of 100,000 images for train, validation, and test sets. All the images have 1640×590 pixels.

LLAMAS [2] is also a large scale lane detection dataset with over 100k images. The lane markers in LLAMAS are automatically annotated with highly-accurate maps. Since the label of test set is not public, we upload the detection result to the website of LLAMAS benchmark for testing.

Tusimple [26] lane detection benchmark is one of the most widely used datasets in lane detection. It contains only highway scenes with 3268 images for training, 358 for validation, and 2782 for testing. All have 1280×720 pixels.

4.2. Implementation details

We adopt the ResNet [6] and DLA [30] as our pre-trained backbones. All input images are resized to 320×800. For data augmentation, similar to [12, 20], we use random affine transformation (translation, rotation, and scaling), random horizontal flips. In the optimizing process, we use AdamW [16] optimizer with an initial learning rate of 1e-3 and cosine decay learning rate strategy [15] with power set to 0.9. We train 15 epochs, 70 epochs, 20 epochs for CULane, Tusimple, and LLAMAS, respectively. Our network is implemented based on Pytorch with 1GPU to run all the experiments. We set the number points of lane prior $N = 72$, and the sampled number $N_p = 36$. The resized H, W in ROIGather are $10, 25$, respectively, channel $C = 64$. The extended radius e in $LIoU$ is 15. The coefficients of assigning cost are set as $w_{cls} = 1$ and $w_{sim} = 3$.

4.3. Evaluation Metric

We adopt the F1-measure as evaluation metric for CULane [17] and LLAMAS [2]. Intersection-over-union (IoU) is calculated between predictions and ground truth. Predicted lanes whose IoU are larger than a threshold (0.5) are considered as true positives (TP). The F_1 is defined as:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

Following COCO [11] detection metric, we also report a new metric mF1 to better compare the localization performance of algorithms. It is defined as

$$\text{mF1} = (\text{F1@50} + \text{F1@55} + \cdots + \text{F1@95})/10,$$

where F1@50, F1@55, $\cdots$, F1@95 are F1 metrics when IoU thresholds are $0.5, 0.55, \cdots, 0.95$ respectively. This is a break from the tradition which will reward detectors with better localization results.

For Tusimple [26] dataset, the evaluation formula is

$$Accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}},$$

where C_{clip}, S_{clip} are the number of correct points and the number of ground truth points of a image respectively. A predicted lane is a correct one if more than 85% predicted lane points are within 20 pixels the ground truth. Tusimple dataset also reports the rate of false positive (FP) and false negative (FN), where $FP = \frac{F_{pred}}{N_{pred}}, FN = \frac{M_{pred}}{N_{gt}}$.

Method	Backbone	mF1	F1@50	F1@75	FPS	GFlops	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	Cross	Night
SCNN [17]	VGG16	38.84	71.60	39.84	7.5	328.4	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10
RESA [33]	ResNet34	-	74.50	-	45.5	41.0	91.90	72.40	66.50	72.00	46.30	88.10	68.60	1896	69.80
RESA [33]	ResNet50	47.86	75.30	53.39	35.7	43.0	92.10	73.10	69.20	72.80	47.70	88.30	70.30	1503	69.90
FastDraw [18]	ResNet50	-	-	-	90.3	-	85.90	63.60	57.00	69.90	40.60	79.40	65.20	7013	57.80
E2E [29]	ERFNet	-	74.00	-	-	-	91.00	73.10	64.50	74.10	46.60	85.80	71.90	2022	67.90
UFLD [19]	ResNet18	38.94	68.40	40.01	**282**	8.4	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10
UFLD [19]	ResNet34	-	72.30	-	170	16.9	90.70	70.20	59.50	69.30	44.40	85.70	69.50	2037	66.70
PINet [7]	Hourglass	46.81	74.40	51.33	25	-	90.30	72.30	66.30	68.40	49.80	83.70	65.20	1427	67.70
LaneATT [24]	ResNet18	47.35	75.13	51.29	153	9.3	91.17	72.71	65.82	68.03	49.13	87.82	63.75	**1020**	68.58
LaneATT [24]	ResNet34	49.57	76.68	54.34	129	18.0	92.14	75.03	66.47	78.15	49.39	88.38	67.72	1330	70.72
LaneATT [24]	ResNet122	51.48	77.02	57.50	20	70.5	91.74	76.16	69.47	76.31	50.46	86.29	64.05	1264	70.81
LaneAF [1]	ERFNet	48.60	75.63	54.53	24	22.2	91.10	73.32	69.71	75.81	50.62	86.86	65.02	1844	70.90
LaneAF [1]	DLA34	50.42	77.41	56.79	20	23.6	91.80	75.61	71.78	79.12	51.38	86.88	72.70	1360	73.03
SGNet [22]	ResNet18	-	76.12	-	117	-	91.42	74.05	66.89	72.17	50.16	87.13	67.02	1164	70.67
SGNet [22]	ResNet34	-	77.27	-	92	-	92.07	75.41	67.75	74.31	50.90	87.97	69.65	1373	72.69
FOLOLane [20]	ERFNet	-	78.80	-	40	-	92.70	77.80	75.20	79.30	52.10	89.00	69.40	1569	74.50
CondLane [12]	ResNet18	51.84	78.14	57.42	173	10.2	92.87	75.79	70.72	80.01	52.39	89.37	72.40	1364	73.23
CondLane [12]	ResNet34	53.11	78.74	59.39	128	19.6	93.38	77.14	71.17	79.93	51.85	89.89	73.88	1387	73.92
CondLane [12]	ResNet101	54.83	79.48	61.23	47	44.8	93.47	77.44	70.93	80.91	54.13	90.16	75.21	1201	74.80
CLRNet (ours)	ResNet18	55.23	79.58	62.21	119/206*	11.9	93.30	78.33	73.71	79.66	53.14	90.25	71.56	1321	75.11
CLRNet (ours)	ResNet34	55.14	79.73	62.11	103/156*	21.5	93.49	78.06	74.57	79.92	54.01	90.59	72.77	1216	75.02
CLRNet (ours)	ResNet101	55.55	80.13	**62.96**	46/74*	42.9	**93.85**	78.78	72.49	82.33	54.50	89.79	**75.57**	1262	**75.51**
CLRNet (ours)	DLA34	**55.64**	**80.47**	62.78	94/151*	18.5	93.73	**79.59**	**75.30**	82.51	54.58	90.62	74.13	1155	75.37

Table 1. State-of-the-art results on CULane. For a fairer comparison, we remeasure the FPS of the source code available detectors using one NVIDIA 1080Ti GPU on the same machine, * means FPS on TensorRT. In addition, we also evaluation these detectors to report the mF1, F1@50, F1@75. For "Cross" category , only false positives are shown. The reported metric of these categories is based on F1@50.

4.4. Comparison with the state-of-the-art results

Performance on CULane. We show the results of our method on the CULane lane detection benchmark dataset and compare them with other popular lane detection methods. As illustrated in Table 1, our proposed method achieves a new state-of-the-art on CULane with an 80.47 F1@50 measure. The ResNet18 version of our method achieves 79.58 F1@50, which is even higher than Cond-LaneNet (ResNet101) while getting 1.4 points higher than CondLaneNet (ResNet18). In particular, we surpass Cond-LaneNet (ResNet18) by 3.4% mF1, which indicates our method better regresses lanes with high localization accuracy. Comparing line anchor-based method LaneATT, our ResNet18 version surpasses 7.88 % mF1 and 4.45 % F1@50, respectively. In the meantime, CLRNet can achieve 206 FPS in one NVIDIA 1080Ti GPU with TensorRT, which is efficient for real-time lane detection.

We show the qualitative results on the CULane dataset in Fig. 6. Segmentation-based methods like RESA don't predict the lane as a whole unit, which can not preserve the smoothness of lanes. CondLaneNet only predicts one start point of the lane as the proposal, it is easy to miss some lane instances. Our method can predict continuous and smooth lanes in these challenging scenarios, which demonstrates our method can definitely gather global context and has a strong ability to detect accurate lanes.

Performance on LLAMAS. The result on the LLAMAS dataset is shown in Table 2. Our method outperforms Poly-LaneNet [25] and LaneATT [24] by 7.7 F1@50 and 2.4

Method	Backbone	valid			test
		mF1	F1@50	F1@75	F1@50
PolyLaneNet [25]	EfficientnetB0	48.82	90.2	45.40	88.40
LaneATT [24]	ResNet18	69.22	94.64	82.36	93.46
LaneATT [24]	ResNet34	69.63	94.96	82.79	93.74
LaneATT [24]	ResNet122	70.8	95.17	84.01	93.54
LaneAF [1]	DLA34	69.31	96.90	84.71	96.07
CLRNet (ours)	ResNet18	**71.61**	96.96	**85.59**	96.00
CLRNet (ours)	DLA34	71.21	**97.16**	85.33	**96.12**

Table 2. State-of-the-art results on LLAMAS. Additionally, we re-run the evaluation for these methods with source code and trained models to get the mF1, F1@50, F1@75.

F1@50 respectively on the test set, which is significant improvement. Although LaneAF [1] achieves 96.90 F1@50 in the valid dataset, its inference speed is slow (near 20FPS), which makes it hard for deployment. Moreover, our method achieves near 2 points mF1 higher than LaneAF, which demonstrates our method is more accurate in localization.

Performance on Tusimple. Table 3 shows the performance comparison with state-of-the-art approaches. The performance difference between different methods on this dataset is very small, which shows the result in this dataset seems to be saturated (high value) already. Our method achieves a new start-of-the-art in terms of F1 score and surpasses the previous state-of-the-art with a 0.6% F1 score. This significant improvement manifests the effectiveness of our method.
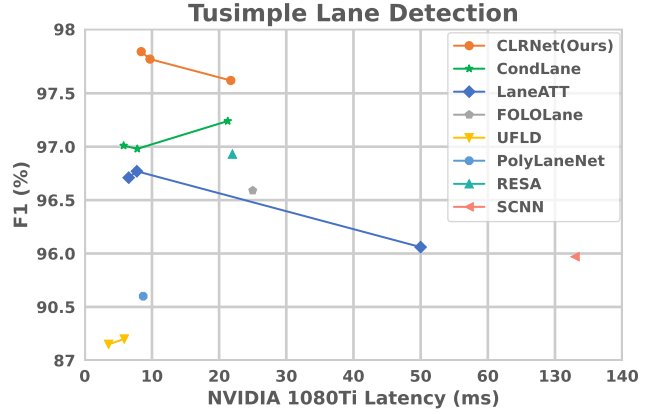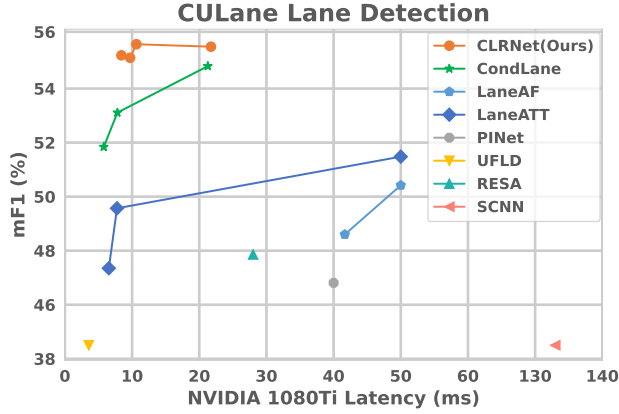
Figure 4. Latency vs. F1-score of state-of-the-art methods on CULane and Tusimple benchmarks.

Method	Backbone	F1 (%)	Acc (%)	FP (%)	FN (%)
SCNN [17]	VGG16	95.97	96.53	6.17	**1.80**
RESA [33]	ResNet34	96.93	96.82	3.63	2.48
PolyLaneNet [25]	EfficientNetB0	90.62	93.36	9.42	9.33
E2E [29]	ERFNet	96.25	96.02	3.21	4.28
UFLD [19]	ResNet18	87.87	95.82	19.05	3.92
UFLD [19]	ResNet34	88.02	95.86	18.91	3.75
LaneATT [24]	ResNet18	96.71	95.57	3.56	3.01
LaneATT [24]	ResNet34	96.77	95.63	3.53	2.92
LaneATT [24]	ResNet122	96.06	96.10	5.64	2.17
FOLOLane [20]	ERFNet	96.59	**96.92**	4.47	2.28
CondLaneNet [12]	ResNet18	97.01	95.48	2.18	3.80
CondLaneNet [12]	ResNet34	96.98	95.37	2.20	3.82
CondLaneNet [12]	ResNet101	97.24	96.54	**2.01**	3.50
CLRNet (ours)	ResNet18	**97.89**	96.84	2.28	1.92
CLRNet (ours)	ResNet34	97.82	96.87	2.27	2.08
CLRNet (ours)	ResNet101	97.62	96.83	2.37	2.38

Table 3. State-of-the-art results on TuSimple. Additionally, F1 was computed using the official source code.

LIoU	Refinement	ROIGather	mF1	F1@50	F1@75	F1@90
			51.90	78.37	58.32	14.43
✓			52.80	78.27	59.50	16.54
✓	✓		54.74	78.91	61.77	20.09
✓	✓	✓	**55.23**	**79.58**	**62.21**	**20.64**

Table 4. Effects of each component in our method. Results are reported on CULane.

4.5. Ablation study

To validate the effectiveness of different components of the proposed method, we conducted several experiments on the CULane dataset to show the performance.

Overall Ablation Study. To analyze the importance of each proposed method, we report the overall ablation studies in Table 4. We gradually add LIoU loss, Cross Layer Refinement, and ROIGather on the ResNet18 baseline. LIoU loss improves the mF1 from 51.90 to 52.80. This result validates that the localization accuracy is much im-
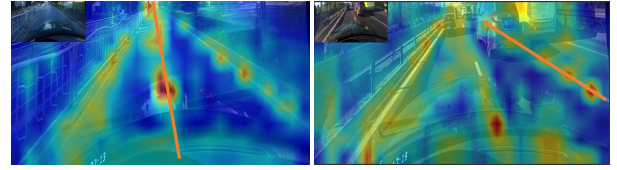

Figure 5. Illustration of attention weight in ROIGather. It shows attention (Eq. 2) between the ROI feature of the lane prior and the whole feature map. The orange line is the correspond lane prior. The red regions corresponds to high score in attention weight.

proved. Moreover, the refinement further improves the mF1 to 54.74. Results in mF1, F1@50, F1@70, and F1@90 are consistently improved, which validates that leveraging high-level and low-level semantic features to detect lanes is useful and yields consistent improvements. ROIGather further improves mF1 by 0.5%, which validates rich global context can enhance the representation of lane features.

Analysis for ROIGather. To further demonstrate how ROIGather works in the network, we visualize the attention map (Eq. 2) in Fig. 5. It shows attention weight between the ROI feature of the lane prior (orange line) and the whole feature map. The brighter the color is, the larger the weight value is. Notably, the proposed ROIGather can (i) effectively gather global context with rich semantic information, (ii) capture the feature of foreground lanes even under occlusion. More quantitative results are in Appendix.

Ablation study on Cross Layer Refinement. Ablation studies of Cross Layer Refinement are shown in Table 5. We first implement the detector with only one layer to perform refinement. As we can see from the result (setting R_0, R_1, R_2), these three refinements get similar results. R_2 gets relatively high F1@90 while the F1@50 is relatively low, indicating low-level features help regress lanes accurately. However, it may cause false detection due to losing high semantic information. We select the better result R_0 and gradually add more refinements. As $R_0 \rightarrow R_0$ shows,
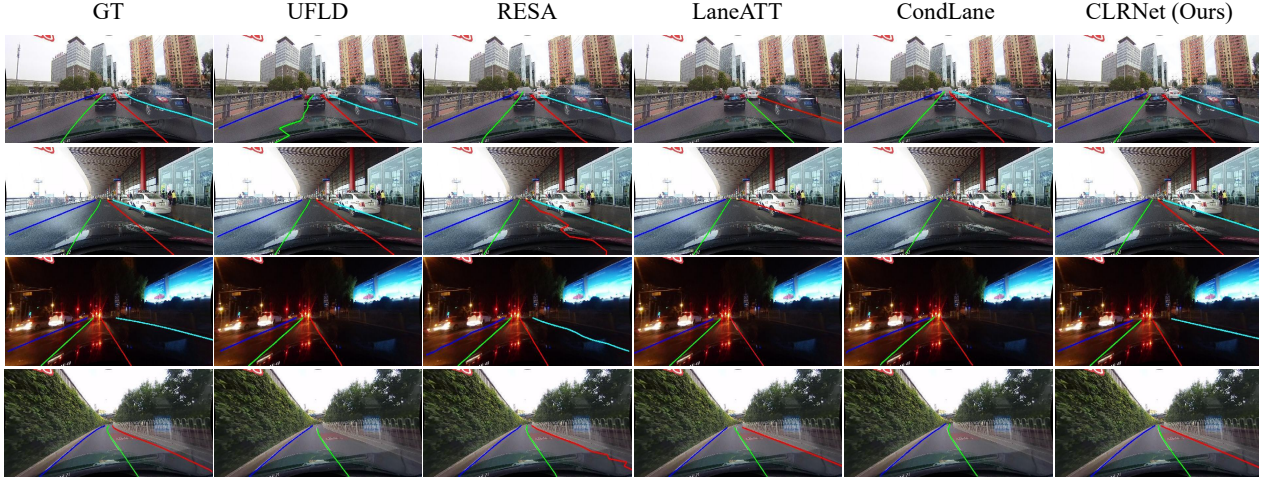
Figure 6. Visualization results of UFLD, RESA, LaneATT, CondLane and our method on CULane testing set.

Settings	mF1	F1@50	F1@75	F1@90
R_0	52.80	78.27	59.50	16.54
R_1	52.26	78.05	59.27	16.71
R_2	52.50	77.82	59.05	16.82
$R_0 \rightarrow R_0$	53.02	78.63	59.74	16.88
$R_0 \rightarrow R_0 \rightarrow R_0$	53.21	78.61	60.11	16.96
$R_2 \rightarrow R_1 \rightarrow R_0$	53.58	78.73	60.44	17.58
ADD	53.77	78.52	60.45	19.01
$R_0 \rightarrow R_1 \rightarrow R_2$	**54.74**	**78.91**	**61.77**	**20.09**

Table 5. Ablation studies of on different refinement methods. R_i is the refinement method discussed in Sec. 3.2. *ADD* means add all features with refinement iteration=3 for a fairer comparison.

it gets slightly improvement. Other fusion feature methods like adding all features still cannot give an improvement. Adopting the refinement from R_0 to R_2 is much better than others, which validates our cross layer refinement can utilize high-level and low-level features better.

Ablation Study on Line-IoU Loss. Ablation studies of Line IoU loss are shown in Table 6. We first turn loss weight to select the best regression weight of smooth-l_1. We observe that the regression loss of smooth-l_1 is much larger than classification loss when regression weight is 1.5. Results show decreasing weight to 0.5 is relatively better. In contrast, LIoU loss is more stable and improves the performance by near 1 point mF1. To be more specific, the improvements mostly come from high overlapping metrics, like F1@80 and F1@90. These experimental results validate that our Line IoU loss can achieve better performance and make the model better converged. We show the proposed Line IoU loss can also improve the performance of LaneATT [24], details can be found in Appendix.

Loss	Weight	mF1	F1@50	F1@60	F1@70	F1@80	F1@90
smooth-l_1	0.1	54.15	79.23	75.00	67.35	51.47	18.98
	0.5	54.22	79.05	74.94	67.21	52.03	19.04
	1.0	53.48	78.33	74.09	66.63	50.85	18.52
	1.5	52.68	78.23	73.57	65.43	49.38	17.58
$\mathcal{L}_{LIoU}$	2	55.23	**79.58**	**75.53**	68.26	53.62	20.64
	4	55.22	79.31	75.43	**68.54**	53.78	20.38
	6	**55.31**	79.35	75.45	68.48	**53.82**	**20.83**

Table 6. Ablation studies of Line IoU loss on CULane.

5. Conclusion

In this paper, we present Cross Layer Refinement Network (CLRNet) for lane detection. CLRNet can exploit high-level features to predict lanes while leveraging local-detailed features to improve localization accuracy. To solve the no visual evidence for the presence of lane, we propose ROIGather to enhance the representation of lane features by building relations with all pixels. To regress lane as a whole unit, we propose Line IoU loss tailored for lane detection, which considerably improves the performance compared with standard loss, *i.e.*, smooth-l_1 loss. Our method is evaluated on three lane detection benchmark datasets, *i.e.,* CULane, LLamas, and Tusimple. Experiments show our proposed method outperforms current state-of-the-art lane detection methods.

6. Acknowledgement

References

[1] Hala Abualsaud, Sean Liu, David Lu, Kenny Situ, Ak-shay Rangesh, and Mohan M Trivedi. Laneaf: Robust multi-lane detection with affinity fields. *arXiv preprint arXiv:2103.12040*, 2021.

[2] Karsten Behrendt and Ryan Soussan. Unsupervised labeled lane marker dataset generation using maps. In *International Conference on Computer Vision (ICCV)*, volume 1, page 7, 2019.

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[4] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[5] 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.

[6] 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.

[7] Yeongmin Ko, Younkwan Lee, Shoaib Azam, Farzeen Munir, Moongu Jeon, and Witold Pedrycz. Key points estimation and point instance segmentation approach for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[8] 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.

[9] 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.

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[12] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Condlanenet: a top-to-down lane detection framework based on conditional convolution. *arXiv preprint arXiv:2105.05003*, 2021.

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

[14] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

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

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

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

[20] Zhan Qu, Huan Jin, Yang Zhou, Zhen Yang, and Wei Zhang. Focus on local: Detecting lane marker from bottom up via key point. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14122–14130, 2021.

[21] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.

[22] Jinming Su, Chao Chen, Ke Zhang, Junfeng Luo, Xiaoming Wei, and Xiaolin Wei. Structure guided lane detection. *arXiv preprint arXiv:2105.05403*, 2021.

[23] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14454–14463, 2021.

[24] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 294–302, 2021.

[25] 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 *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6150–6156. IEEE, 2021.

[26] TuSimple. Tusimple benchmark. https://github.com/TuSimple/tusimple-benchmark/, Accessed September, 2020.

[27] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the*

IEEE conference on computer vision and pattern recognition, pages 7794–7803, 2018.

[28] 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 *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 689–704. Springer, 2020.

[29] 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/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1006–1007, 2020.

[30] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[31] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016.

[32] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[33] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. Resa: Recurrent feature-shift aggregator for lane detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3547–3554, 2021.

[34] Tu Zheng, Shuai Zhao, Yang Liu, Zili Liu, and Deng Cai. Scaloss: Side and corner aligned loss for bounding box regression. *arXiv preprint arXiv:2104.00462*, 2021.