Lane Detection with Versatile AtrousFormer and Local Semantic Guidance

Jiaxing Yang[a], Lihe Zhang[a,**], Huchuan Lu[a]

[a]*School of Information and Communication Engineering, Dalian University of Technology, Dalian, 116023, China*

Abstract

Lane detection is one of the core functions in autonomous driving and has aroused widespread attention recently. The networks to segment lane instances, especially with bad appearance, must be able to explore lane distribution properties. Most existing methods tend to resort to CNN-based techniques. A few have a try on incorporating the recent adorable, the seq2seq Transformer [1]. However, their innate drawbacks of weak global information collection ability and exorbitant computation overhead prohibit a wide range of the further applications. In this work, we propose Atrous Transformer (AtrousFormer) to solve the problem. Its variant local AtrousFormer is interleaved into feature extractor to enhance extraction. Their collecting information first by rows and then by columns in a dedicated manner finally equips our network with stronger information gleaning ability and better computation efficiency. To further improve the performance, we also propose a local semantic guided decoder to delineate the identities and shapes of lanes more accurately, in which the predicted Gaussian map of the start-

*Corresponding author

**

ing point of each lane serves to guide the process. Extensive results on three challenging benchmarks (CULane, TuSimple, and BDD100K) show that our network performs favorably against the state of the arts.
Keywords: Lane Detection, Local AtrousFormer, Global AtrousFormer, Local Semantic Guided Decoder.

1. Introduction

Autonomous driving becomes increasingly promising recently. In such a systematic engineering project, one core desirement is to generate high-quality lane instances (both online and offline). The vehicles rely on the generated results to keep itself from overstepping the boundaries [2]. However, lane detection faces big challenges from the wild scenes (see Fig. 1). Lanes may be ambiguous due to occlusion by the vehicles, bad weather conditions, abrasion, illumination intensity, and etc. In addition, the task also suffers from the disconnected, curved and slender shapes of lane itself. To solve the problems, traditional methods [3, 4, 5, 6, 7, 8, 9, 10] tend to adopt a two-stage strategy. They first use hand-crafted operators to extract the feature, and then apply an adjustment step in order to accommodate the real line shape, such as Hough transform [3, 4] and Random Sampling [6, 7]. Nevertheless, although some progresses have been made, these methods are not good enough to handle complex traffic scenes.

Recently, as deep learning based techniques gradually dominates various computer vision tasks, community begins to focus on the CNN-based techniques to facilitate the research of lane detection. Most methods define lane detection as a pixel-level semantic segmentation problem [11, 12,

Figure 1: Visualization of lanes in bad cases: Occlusion, Crowded, High Light, and Arrow Shape.

13, 14, 15, 16, 17, 18]. Specifically, in these methods, each lane is regarded as a semantic class. They really rely on a strong feature to infer the ideal shapes. Another group of them draw on the recent achievements in object detection. They [19, 20, 21, 22] are developed towards two directions: anchor-based [19, 20] and anchor-free [21, 22]. The anchor-based ones use beaming lines starting from either left, right, or bottom as anchor baseline, and then adjust offset to fit the real shape of lane line via a post-processing step. They heavily rely on pre-defined straight lines, thus struggling to handle more complex line shapes. As for those anchor-free, they equate lane detection to high-order polynomial regression, straightforward yet overly relying on certain parameters. The last main group [23, 24, 25, 26], inspired by the fancy thoughts from human pose estimation, usually extract key points with lane semantic and then cluster them into different lane instances via complex post-processing methods. In general, methods other than sementic segmentation have difficulties in modeling more complex lane forms, like

those described in BDD100K [27].

In this paper, we walk further along the way of semantic segmentation based lane detection. Hereafter we get a closer look on its members. Early of them such as [12, 11, 17, 13] have a CNN-based encoder, a plain upsampling decoder, and a binary classifier. However, their quantitative and visualization results indicate that the segmented lanes are dragged down by low-quality feature map. The problem is generated due to their ignorance towards the spatial cues of lanes and incapability of conjecturing according to the environment information. To overcome, recent explorers such as SCNN [14] and RESA [18] propagate information between near or remote feature slices in four directions (upwards, downwards, rightwards, and leftwards). However, recurrent reinforcement manner does not treat features equally and therefore does not perform well (similar to LSTM). More advanced way still needs exploring. Another problem haunting the prediction of lane instances is their weak discrimination ability, that is to say, the classifiers feeding on global representation blunder in predicting the existence of lanes.

Instead of focusing on the temporally-recurrent CNN-based techniques, in this paper, we resort to designing Atrous Transformer (AtrousFormer). In the same spirit as ASPP [28], the AtrousFormer instills atrous experience to the transformer structure [1] by following a two stage strategy. It collects information first along rows and then along columns. Compared to [1], our designing can save lots of computation cost (has reduced the number of key&value entities) and improve performance, simultaneously. Note that AtrousFormer has two forms, namely global AtrousFormer and local AtrousFormer. The global one is installed on top of the feature extractor to enhance

global representation. The local one is used to implement early extracting enhancement, in a local manner. We also propose Local Semantic Guided Decoder (LSGD), in which the Gaussian map of the starting point of each lane is employed as the attention map to guide classification process (the distance of each other in this position is long enough). Our contributions can be boiled down into the following points:

- We propose global AtrousFormer to collect information in an atrous yet global way, instilling the essence of the ASPP to the transformer structure and finally enhancing the ability to infer lane distribution.
- Global AtrousFormer is evolved into a more compact version, local AtrousFormer, by incorporating slice mechanism. Then it is early embedded into feature extractor like ResNet-18 to enhance feature extraction.
- We propose LSGD to obtain more representative feature vector, which then is used to better the lane existence prediction.
- Our network achieves favorable results against other state of the arts on the recent challenging datasets, achieving 78.08 F1 score on CULane [14] and 96.71 Accuracy on TuSimple [29].

2. Related Works

This section will discuss first the deep learning based lane detection methods. In general, the networks to conduct lane detection can be categorized into three classes, semantic segmentation based methods, detection based

methods, and key point estimation based methods. Finally, the development of transformer in vision community is sketched.

Semantic Segmentation Based Methods. In this kind of doings, lane detection is modeled as a per-pixel prediction task with an additional classification branch [11, 12, 13, 14, 15, 16, 17, 18]. To improve performance, early they devise various enhancing mechanisms [11, 12, 13, 17, 15] and recently resort to feature aggregator modules [14, 18]. The [15] uses the method of self-distillation to mine spatial properties of lanes. The [12] designs a new powerful backbone for lane detection. The authors in [14] use a in-layer, recurrent in four directions, and residual information passing mechanism to enhance the spatial feature representation, finally accommodating the thin and long lane shapes. Later, [18] mitigates these problems in a sparse gleaning way.

Detection Based Methods. Motivated by the progresses in recent object detection based methods, lane detection research in this direction [19, 20, 21, 22] can be classified as anchor-based [19, 20] and anchor-free [21, 22]. The [19, 20] use beaming lines from either the bottom, left, right to propose primary candidates, and then predicts offset maps to fit the real line shape. However, they are struggling to free themselves from the constraints of anchors. In [20], local cues are added to refine the problem. The works of [21, 22] abandon this kind of doing and directly assume that lanes in the scenes can be modeled as a polynomial equation estimation problem. Although refreshing, they are infeasible to handle the complex scenes, due to overly relying on certain parameters.

Key Point Estimation Based Methods. This approach [23, 24, 25,

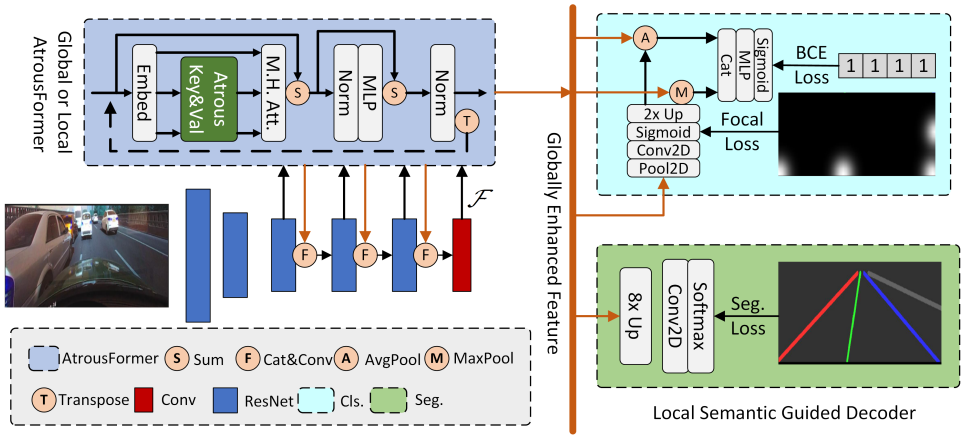

Figure 2: An overview of the architecture. It passes the raw image through local AtrousFormer Enhanced extractor, one 1×1 Convolution, Global AtrousFormer, and Local Semantic Guided Decoder in sequence, finally generating segmentation maps and the corresponding classification scores.

26] usually determines the sparse or dense key points belonging to lanes first via a branch, and simultaneously in other branches predicts their identities to cluster them into different instances. In [24], the authors design a branched, multi-task network, including a binary segmentation branch and a identity embedding branch, to clustering key points into different lane instances in an end to end manner. However, in most cases it is suffered from occlusion problem. Therefore, [24] stack several hourglass modules to enhance the inference ability of the network to alleviate the problem. Recently, [25] proposes to use two affinity matrices to cluster the determined key points to further resolve the problem.

Vision Transformers. The seq2seq Transformer[1] is primarily proposed to resolve the problems long haunting around LSTM and various its posteriors, such as long-range information losing, weak semantic represen-

tation, and sequential inflexibility. Recently, researchers begin introducing it into vision community to enhance the feature representation of backbone. The Vision Transformer [30] is the primary one applying transformer structure on non-overlapping medium-size feature patches for image classification, to some degree achieving speed-accuracy balance. Later, one of its follow-ups swin transformer [31] uses window and shifting mechanisms at different scales, further alleviating the overdue computational problem and achieving good performance in different tasks. Others like [32, 33] tinker it in various ways.

3. Methodology

In this section, we show the overall architecture of our network (see Fig.2), the designed global AtrousFormer, local AtrousFormer enhanced extractor and Local Semantic Guided Decoder (LSGD).

3.1. Overall Architecture

We first extract feature using local AtrousFormer enhanced extractor (enhanced ResNet-18, ResNet-34, etc). The max poolings in the third and fourth stages are replaced with dilations. After the raw image passes through the extractor, a feature with spatial size of 1/8 original image is generated. Its channel dimension followingly is compressed to C^{cmpr} by one 1×1 convolution, denoted as $\mathcal{F}$ of size $\mathrm{H} \times \mathrm{W} \times \mathrm{C}^{cmpr}$. We then send $\mathcal{F}$ to global AtrousFormer to further enhance the semantic representation, and the enhanced feature is finally fed to the decoder LSGD. The decoder consists of a segmentation branch and a classification branch. In the segmentation branch, the spatially enhanced feature is directly recovered to the original

size by 8× bilinear upsampling. And in the sequel, the product is processed by a convolution manipulation to generate the final segmentation maps (the maps later are used to predict the distribution of lane instances). In the classification branch, we primarily get the Gaussian map of the starting point of each lane. Then the map is used to do spatial attention on the enhanced feature. Finally we obtain representative feature vector of each lane to clarify the existence of lanes. In the following context, the global AtrousFormer is first introduced for narrating consistency.

3.2. Global AtrousFormer

Let us at the beginning have a retrospect on the seq2seq Transformer. It fuses the thoughts of position-aware embedding, multi-head self- and cross-attention mechanism, and residual connection, making the extracted features more expressive. However, in lane detection, lane itself is slender, and its spreading direction in either the real world or on the image plane follows certain geometric and human-setting rules. The ignorance of the seq2seq Transformer towards this observation not only causes computational overhead to surge, but also introduces more noises.

The global AtrousFormer imparts the prior knowledge to the seq2seq Transformer, decomposing the sampling process into a two-stage atrous form. The process of the first stage is formulated in the next. The feature $\mathcal{F}$ extracted by the backbone is embedded into three tensors:

$$\begin{aligned} \mathcal{Q}^{global} &= ConvQ_{1\times1}(\mathcal{F}+\mathcal{P}), \\ \mathcal{E}^{key} &= ConvK_{1\times1}(\mathcal{F}+\mathcal{P}), \\ \mathcal{E}^{val} &= ConvV_{1\times1}(\mathcal{F}). \end{aligned} \tag{1}$$

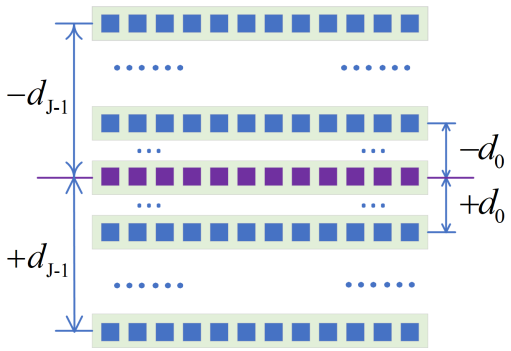

Figure 3: The positions of query entities (in purple boxes) and key&value entities (in both blue and purple boxes) in global AtrousFormer.

In (1), $ConvQ_{1\times1}$ refers to 1×1 convolution manipulation with stride and padding size equivalent to 1 and 0, respectively. Its output $\mathcal{Q}^{global}$ is a tensor of size $\mathrm{H}\times\mathrm{W}\times\mathrm{C}^{cmpr}$ and can be directly used as the query of the global AtrousFormer. The other two operations $ConvK_{1\times1}$ and $ConvV_{1\times1}$ denote 1×1 convolutions with stride of 1 and padding size of $\mathrm{H}\times0$ (row and column wise paddings). Note that the padding serves only as placeholder that helps the generation of affinity matrices. Both $\mathcal{E}^{key}$ and $\mathcal{E}^{val}$ are of the same size $3\mathrm{H}\times\mathrm{W}\times\mathrm{C}^{cmpr}$, serving to generate the final key $\mathcal{K}^{global}$ and value $\mathcal{V}^{global}$. The positional encoding tensor $\mathcal{P}$ of size $\mathrm{H}\times\mathrm{W}\times\mathrm{C}^{cmpr}$ is generated using the same method in [1].

Next we introduce the production of $\mathcal{K}^{global}$ and $\mathcal{V}^{val}$. In global AtrousFormer, members of $\mathcal{Q}^{global}$ in the ith row will query information from those that are not only in the ith row, but also those above and below the ith row in the distances of $\{d_j\}_{j\in\forall}$ (see Fig.3):

$$d_j = \lfloor \mathrm{H}/2^{(\mathrm{J}-j)} \rfloor, \quad j = 0, 1, ..., \mathrm{J}-1, \tag{2}$$

where J is a hyper-parameter specifying the sampling density and $d_j \geq 1$.

The $\lfloor - \rfloor$ rounds down a float number to its closest integer. The $\mathcal{K}^{global}$ therefore can be generated using the following pattern:

$$\begin{aligned}\mathcal{K}^{global} = Cat(&\mathcal{E}^{key}[\mathrm{H}:2\mathrm{H},:,:],\\ &\mathcal{E}^{key}[\mathrm{H}\pm d_0:2\mathrm{H}\pm d_0,:,:]\\ &,...,\\ &\mathcal{E}^{key}[\mathrm{H}\pm d_{\mathrm{J}-1}:2\mathrm{H}\pm d_{\mathrm{J}-1},:,:]).\end{aligned} \tag{3}$$

In (3), we resort to python conservation word $a:b$ to denote index range from a to b with the default interval of 1. The Cat concatenates the tensors along the second dimension. The output $\mathcal{K}^{global}$ is of size $\mathrm{H}\times(2\mathrm{J}+1)\mathrm{W}\times\mathrm{C}^{cmpr}$. By replacing $\mathcal{E}^{key}$ with $\mathcal{E}^{value}$ in (3), we will get $\mathcal{V}^{global}$ of the same size as $\mathcal{K}^{global}$. The heuristic experience is very effective considering the disconnected, thin, and forward-spreading distribution properties of lane instances. More members in the neighboring region are queried than those in the remote area.

Split $\mathcal{Q}^{global}$, $\mathcal{K}^{global}$, and $\mathcal{V}^{global}$ into N^{heads} heads along the channel dimension, each of which has $\mathrm{C}^{cmpr}/\mathrm{N}^{heads}$ feature maps. As such, the affinity matrices can be computed as follows:

$$\mathcal{A}^{global} = \frac{Softmax(\mathcal{Q}^{global}@Perm(\mathcal{K}^{global}))}{\sqrt{\mathrm{C}^{cmpr}/\mathrm{N}^{heads}}}, \tag{4}$$

in which $Perm$ means to exchange the index order of the second and third dimension, $Softmax$ represents the row-wise normalization, and the symbol @ denotes matrix multiplication. The produced affinity matrices $\mathcal{A}^{global}$ is of size $\mathrm{N}^{heads}\times\mathrm{H}\times\mathrm{W}\times(2\mathrm{J}+1)\mathrm{W}$. Similar to the seq2seq Transformer, we

get the final output by using the following manipulations:

$$\begin{aligned}\mathcal{I}^{global} &= Norm(\mathcal{F} + Recover(\mathcal{A}^{global}@\mathcal{V}^{global}),\\ \mathcal{F}^{global} &= Norm(\mathcal{I}^{global} + MLP(\mathcal{I}^{global})).\end{aligned} \tag{5}$$

In (5), $Norm$ and MLP refer to layer normalization and multiple layer perceptrons, respectively. The *Recover* consisting of consecutive permute and reshape manipulations is used to recover the multi-head feature into the original size. As of now, how members in each row glean information by the proposed atrous way is introduced. The second stage is quite similar to the first stage, first exchanging the row and column indices of $\mathcal{F}^{global}$, then passing it through the equations from (1) to (5), and finally producing the global AtrousFormer enhanced feature.

3.3. Local AtrousFormer for Early Enhancement

We further consider interleaving the global AtrousFormer into the early feature extraction stage to enhance the inference ability of our network. However, directly inserting the global AtrousFormer into different stages of ResNet will be costly in terms of hardware and computation overhead. Instead we develop local AtrousFormer by introducing rectangular box restriction to the global AtrousFormer, but in the meanwhile take the spatial distribution attributes of lanes into account.

We apply the local AtrousFormer to the second, third and fourth stages, all of which generate representative features of size 1/8 original image (replacing the max poolings in the third and fourth stages with dilated convolution of atrous rates of 2 and 4, respectively). The unenhanced feature after each stage is sent to the local AtrousFormer, and then concatenated with the

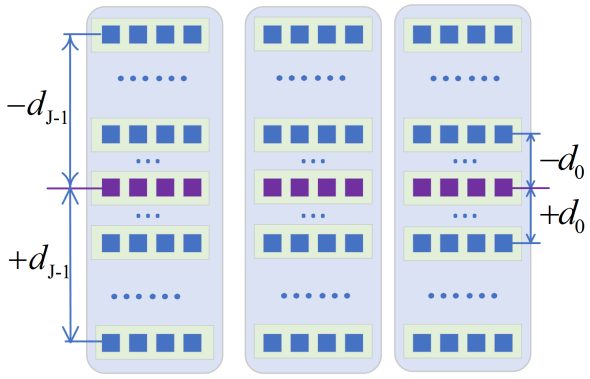

Figure 4: The positions of query entities (in purple boxes) and key&value entities (in both blue and purple boxes) in local AtrousFormer. The gleaning process happens within each slice.

production to fuse to go to the next stage. This kind of multi-scale fusion objectively strengthens the local information communication, just like the window shifting mechanism in [31].

Next we give the details of local AtrousFormer in the ith stage ($i \in \{2, 3, 4\}$), which also has gleaning information process of two stages. During the first stage we split the primary feature into S^{col} slices along column, and do the sparse attention only within the slice. After the tensors $\mathcal{Q}_i^{local}$, $\mathcal{E}_i^{key}$, and $\mathcal{E}_i^{val}$ are generated by (1), we transform them into size of $\mathrm{H} \times \mathrm{S}^{col} \times \mathrm{W}/\mathrm{S}^{col} \times \mathrm{C}_i^{stage}$, $3\mathrm{H} \times \mathrm{S}^{col} \times \mathrm{W}/\mathrm{S}^{col} \times \mathrm{C}_i^{stage}$, and $3\mathrm{H} \times \mathrm{S}^{col} \times \mathrm{W}/\mathrm{S}^{col} \times \mathrm{C}_i^{stage}$ using the *Reshape* operation, respectively. Probing heuristic experience in equation (2) is still adopted in the similar pattern. The key tensor can be

generated (see Fig.4) as follows:

$$\begin{aligned}\mathcal{K}_i^{local} = Cat(&\mathcal{E}_i^{key}[\mathrm{H}:2\mathrm{H},:,:,:],\\ &\mathcal{E}_i^{key}[\mathrm{H}\pm d_0:2\mathrm{H}\pm d_0,:,:,:]\\ &,...,\\ &\mathcal{E}_i^{key}[\mathrm{H}\pm d_{\mathrm{J}-1}:2\mathrm{H}\pm d_{\mathrm{J}-1},:,:,:]),\end{aligned} \tag{6}$$

where the operation of Cat will concatenate the tensors along the third dimension, and other manipulations are similar to those previously introduced. The output $\mathcal{K}_i^{local}$ is of size $\mathrm{H}\times\mathrm{S}^{col}\times(2\mathrm{J}+1))\mathrm{W}/\mathrm{S}^{col}\times\mathrm{C}_i^{stage}$. The value tensor $\mathcal{V}_i^{local}$ of size $\mathrm{H}\times\mathrm{S}^{col}\times(2\mathrm{J}+1))\mathrm{W}/\mathrm{S}^{col}\times\mathrm{C}_i^{stage}$ also can be produced by replacing $\mathcal{E}_i^{key}$ with $\mathcal{E}_i^{val}$ in (6).

Split features into N_i^{heads} heads and the affinity matrices for the local AtrousFormer can be computed in the following way:

$$\mathcal{A}_i^{local} = \frac{Softmax(\mathcal{Q}_i^{local}@Perm(\mathcal{K}_i^{local}))}{\sqrt{\mathrm{C}_i^{stage}/\mathrm{N}_i^{heads}}}, \tag{7}$$

in which the $Perm$ manipulation is used to exchange the third and fourth dimension of $\mathcal{K}_i^{local}$. The output $\mathcal{A}_i^{local}$ is of size $\mathrm{N}_i^{heads}\times\mathrm{H}\times\mathrm{S}^{col}\times\mathrm{W}/\mathrm{S}^{col}\times(2\mathrm{J}+1)\mathrm{W}/\mathrm{S}^{col}$. Applying equation (5), we will get the locally row-wise enhanced feature. The second stage of the local AtrousFormer is first to transpose the row and column indices of input and the rest process is similar to what we have introduced in the end of the last subsection. Compared to attention mechanism in seq2seq Transformer, whose computation of self-attention involves $(\mathrm{HW})^2$ dot products, the computation in global AtrousFormer and local AtrousFormer have $(\mathrm{HW})*(2\mathrm{J}+1)(\mathrm{H}+\mathrm{W})$ and

$(\mathrm{HW}) * (2\mathrm{J}+1)(\mathrm{H}/\mathrm{S}^{row} + \mathrm{W}/\mathrm{S}^{col})$, respectively. The S^{row} denotes the slice number along row wise.

3.4. Local Semantic Guided Decoder

The methods based on semantic segmentation tend to directly apply fully connected layers coupled with *Pooling* to clarify the existence of each lane in one branch, and $Conv2D$ layers to segment the shapes in the other branch. Both of them are on top of the semantically enhanced feature aggregator. Although the designed feature aggregator is strong enough to handle the segmentation subtask, they ignore the negative influence on the current existence prediction from that of others. Differently from them, we adopt predicting the Gaussian map of the starting point of each lane to guide the classification process, which is named as Local Semantic Guided Decoder (LSGD).

Next we will introduce the details of LSGD. In lane detection, the number of lanes is a prior knowledge specified by the annotation rules, just like 4 in CULane and 6 in TuSimple. Therefore, LSGD is able to predict the Gaussian map of the starting point of each lane as follows:

$$\mathcal{G}^{map} = Sig((Conv(MaxPool_{2d}(\mathcal{F}^{global})))), \tag{8}$$

in which $Conv$ of consists of two consecutive 1 stride convolutions with kernels of size $3 \times 3 \times \mathrm{C}^{cmpr} \times \mathrm{C}^{cmpr}$ and $1 \times 1 \times \mathrm{C}^{cmpr} \times \mathrm{N}^{lanes}$ (number of lanes), respectively, $MaxPool_{2d}$ represents $2\times$ maxpooling, and Sig represents sigmoid manipulation. The output of size $\mathrm{N}^{lanes} \times \mathrm{H}/2 \times \mathrm{W}/2$ is supervised by the focal loss in [34]. The ith Gaussian map $\mathcal{G}^{map}_{[i,:,:]}$ serves as the attention

map to guide the classification process of the ith lane:

$$c_i^{score} = Sig(MLP(Cat(MaxPool(\mathcal{F}_i^{attended}), \\ AvgPool(\mathcal{F}_i^{attended})))), \tag{9}$$

in which $\mathcal{F}_i^{attended} = Up_2(\mathcal{G}_{[i,:,:]}^{map}) \otimes \mathcal{F}^{global}$, where Up_2 is bilinear interpolation of $2\times$ upsampling; $MaxPool$ and $AvgPool$ refer to the max and average poolings, respectively; the MLP is three fully connected layers with hidden dimension as 64 and 16. The segmentation maps can be obtained by passing $\mathcal{F}^{global}$ through a $8\times$ bilinear interpolation and a convolution manipulation with stride 1 and kernel size $3 \times 3 \times \mathrm{C}^{cmpr} \times \mathrm{N}^{lanes}$.

4. Experiments

4.1. Datasets

To verify the effectiveness of our proposed methods, we demonstrate their respective performance on three currently wide-used datasets, CULane [14], TuSimple [29], and BDD100 [27]. They are challenging and encompass a variety of scenarios.

To be specific, CULane has 55 hours video clips, and consists of nine road conditions: normal, crowd, curve, dazzle night, night, no line, and arrow. Most of the frames within are shot in the urban, and a few are in the rural or highway. In total, CULane has 133235 frames of size 590×1640, of which 88880 are used for training, 9675 are used to validate, and the left are used to benchmark the performance of the model. The number of lanes in each frame is at most 4.

The samples in TuSimple are shot under stable light situation by vehicles in highways. Compared to CULane, the dataset is relatively small. In total, it has 6408 frames. Of them, 3236 are used for training, 358 are used for validation, and 2782 are used to test. All are of size 720×1280, and each frame contains at most 5 lanes.

The BDD100K originally designed for lane classification has binary annotation for lane semantics (provided by [15]). The lanes are close to each other, and also embody in different forms, thus challenging to current algorithms. Different from CULane and TuSimple, the width of each lane in the training set is set to 8, while in the testing set it is set to 2. Following [15], we use the training set of 80000 frames of resolution 720×1680 to train our model, and validation set of 10000 frames of the same resolution to test.

4.2. Implementation Details

We resize the original images to 288×800 for CULane and 368×640 for TuSimple, respectively. Furthermore, our training involves the following techniques of data augmentation: random scaling, cropping, random rotation, color jittering, and etc. The optimizer uses SGD with momentum 0.9 and weight decay 1e-4 to train our model. The learning rate is set to 2.5e-2 for CULane and 2.0e-2 for TuSimple, respectively. Warming-up strategy is used in the first 500 batches. Polynomial learning rate decay policy with power set to 0.9. The loss function is the summation of the focal loss multiplied by 0.2 and the one in [14, 18], which consists of segmentation BCE loss and existence classification BCE loss. The batch size is set to 8 for CULane and 4 for TuSimple. The number of training epoch is set to 12 for CULane and 80 for TuSimple. All models are trained on 3 NVIDIA 2080Ti

GPUs with Pytorch framework. For local AtrousFormer enhanced backbone, we set $\mathrm{N}^{\mathrm{heads}}$ in the second, third, and fourth stages to 2, 8, and 16, respectively, let J equal to 4, and set slice size to $(18, 20)$ and $(23, 20)$ on CULane and TuSimple, respectively. The feature extracted from the backbone are compressed to 128 using 1×1 convolution. For global AtrousFormer, we set J to 4 and N^{heads} to 16. The threshold values are set to 0.5 in both the classification branch and segmentation branch. The settings for BDD100K is slightly different from what we have adpted in CULane. Following [15], we resize the training image to 360×720 in training and use the unresized image to test. During the experiments on BDD100K, the branch of local semantic guidance is removed.

For CULane, lanes are treated as a line of 30-pixel-width. The intersection-over-union (IOU) of the predictions and groundtruth lanes is used. In general, the predicted lanes that have a IOU with their corresponding targets larger or equivalent to the threshold value 0.5 are defined as the true positives (TP). The FP and FN are used to denote false positives and false negatives, respectively. The F1-measure is adopted by CULane to evaluate the results, as follows:

$$\mathrm{F}_1 = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \tag{10}$$

where $\mathrm{Precision} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$ and $\mathrm{Recall} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$.

The TuSimple evaluates the results by Accuracy, defined as follows:

$$\mathrm{Accuracy} = \sum\nolimits_{clip} \frac{\mathrm{C}^{clip}}{\mathrm{S}^{clip}}, \tag{11}$$

in which C^{clip} represents the number of correctly predicted lane points. To be specific, they are assigned the right identities. Their distances with the

corresponding ground truth points are within certain range in a clip. The S^{clip} denotes the number of ground truth points in a clip.

For BDD100K, following [15], we use per-pixel accuracy and mean IOU to evaluate the performance of different models.

4.3. Comparison with the SOTAs

In this section, we will demonstrate the performance of our network on CULane, TuSimple, and BDD100K, respectively. For CULane, the quantitative and visualization results under nine scenarios are showed in Tab.1 and Fig.5, respectively. The quantitative results for TuSimple and BDD100K are listed in Tab.2 and Tab.3. Algorithms to compare include FastDraw [35], SpinNet [36], UFAST [37], R18-E2E [38], R34-E2E [38], ERFNet-E2E [38], SCNN [14], ENet-SAD [15], ERFNet-IntRA-KD [40], PINet [24], Curvelanes-NAS [26], RESA [18], LaneATT [20], and SIM-CycleGan [13]. All of the results are borrowed from their official publishings.

CULane Results Analysis. In Tab.1, the XR18-Ours and XR34-Ours are used to represent implementations based on enhanced ResNet-18 and ResNet-34, respectively. The R18-Light represents the one of ResNet18+Local AtrousFormer (slice size (18, 20))+LSGD. It can be seen that overall our network achieves 70.03, 77.63, and 78.08 scores in terms of F1 metric. Our encoder-enhanced network (XR18-Ours and XR34-Ours) surpasses the current state of the art LaneATT by 2.54 and 1.4 points on ResNet-18 and ResNet-34, respectively. As for those adopting temporally-recurrent based segmentation methods on ResNet34, SCNN and RESA, XR34-Ours tops them by 3.58 and 6.48 points, respectively. For different environments of Normal, Crowded, Dazzle, Shadow, NoLine, Arrow, Curve, Cross and Night,

Table 1: Quantitative results of our network versus other state of the arts under nine situations of CULane test set against the F1 metric. The Cross only reports FP. Note that red, blue, and green colors represents the highest, second, and third scores, respectively. The symbol of '-' represents the results are not reported officially.

Methods	Normal ↑	Crowded ↑	Dazzle ↑	Shadow↑	NoLine ↑	Arrow↑	Curve↑	Cross↓	Night↑	Total↑	Speed (ms)
Fast Draw [35]	85.90	63.60	57.00	59.90	40.60	79.40	65.20	7013	57.80	-	11
SpinNet [36]	90.50	71.70	62.00	72.90	43.20	85.00	50.70	-	68.10	74.20	-
R18-UFAST [37]	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10	68.40	**3**
R34-UFAST [37]	90.70	70.20	59.50	69.30	44.40	85.70	69.50	2037	66.70	72.30	6
R18-E2E [38]	90.00	69.70	60.20	62.50	43.20	83.20	70.30	2296	63.30	70.80	-
R34-E2E [38]	90.40	69.90	61.50	68.10	45.00	83.70	69.80	2077	63.20	71.50	-
ERFNet-E2E [38]	91.00	73.10	64.50	74.10	46.60	85.80	71.90	2022	67.90	74.00	-
R34-SCNN [14]	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10	71.60	116
ERFNet-SAD [15]	90.10	68.80	60.20	65.90	41.60	84.00	65.70	1998	66.00	70.80	10
ERFNet-IntRA-KD	-	-	-	-	-	-	-	-	-	72.40	-
PINet [24]	90.30	72.30	66.30	68.40	49.8	83.70	65.60	1427	67.70	74.40	40
SIM-CycleGAN[13]	91.80	71.80	66.40	76.20	46.10	87.80	67.10	2346	69.40	73.90	-
CurveLanes-NAS-S [26]	88.30	68.60	63.20	68.00	47.90	82.50	66.00	2817	66.20	71.40	-
CurveLanes-NAS-M [26]	90.20	70.50	65.90	69.30	48.80	85.70	67.50	2359	68.20	73.50	-
R34-RESA [18]	91.90	72.40	66.50	72.00	46.30	88.10	68.60	1896	69.80	74.50	22
R18-LaneATT [20]	91.11	72.96	65.72	70.91	48.35	85.49	63.37	1170	68.95	75.09	**4**
R34-LaneATT [20]	92.14	75.03	66.47	**78.15**	49.39	**88.38**	67.72	1330	70.72	76.68	**6**
R122-LaneATT [20]	91.74	**76.16**	**69.47**	**76.31**	**50.46**	86.29	68.40	1746	68.90	77.02	45
R18-Light	**92.77**	74.69	66.89	69.68	49.25	88.09	**70.59**	**1096**	**72.85**	**77.03**	20
XR18-Ours	**92.72**	**75.56**	**68.16**	73.67	**50.07**	**88.82**	**70.32**	**1169**	**73.49**	**77.63**	36
XR34-Ours	**92.83**	**75.96**	**69.48**	**77.86**	**50.15**	**88.66**	**71.14**	**1054**	**73.74**	**78.08**	44

Table 2: Quantitative comparison results with other state of the arts on the test set of TuSimple in terms of Accuracy, FP, and FN metrics.

Methods	Accuracy ↑	FP↓	FN↓
SCNN [14]	96.53	6.17	**1.80**
EL-GAN [17]	94.90	4.12	3.36
PINet [24]	**96.70**	2.94	**2.63**
ENet-SAD [15]	96.64	6.02	**2.05**
ERF-E2E [38]	96.02	3.21	4.28
FastDraw [35]	95.20	7.60	4.50
R18-UFAST [37]	95.82	19.05	3.92
R34-UFAST [37]	95.86	18.91	3.75
PolyLaneNet [21]	93.36	9.42	9.33
LSTR [22]	96.18	2.91	3.38
R18-RESA [18]	**96.82**	3.95	2.83
R34-RESA [18]	**96.70**	3.96	2.48
R18-LaneATT [20]	95.57	3.56	3.01
R34-LaneATT [20]	95.63	3.53	2.92
XR18-Ours	96.59	**2.83**	3.26
XR34-Ours	**96.71**	**2.82**	3.24

Table 3: Quantitative comparison results with other state of the arts on the test set of BDD100K in terms of average per-pexel accuracy and IOU of lane semantic of each frame.

Methods	Accuracy ↑	IOU↑
ResNet18 [39]	30.66	11.07
ResNet34 [39]	30.92	12.24
SCNN [14]	35.79	15.04
R18-SAD [15]	31.10	13.29
R34-SAD [15]	32.68	14.56
ERFNet-SAD [15]	**36.56**	**16.02**
XR18-Ours	**59.09**	**22.81**
XR34-Ours	**59.20**	**23.31**

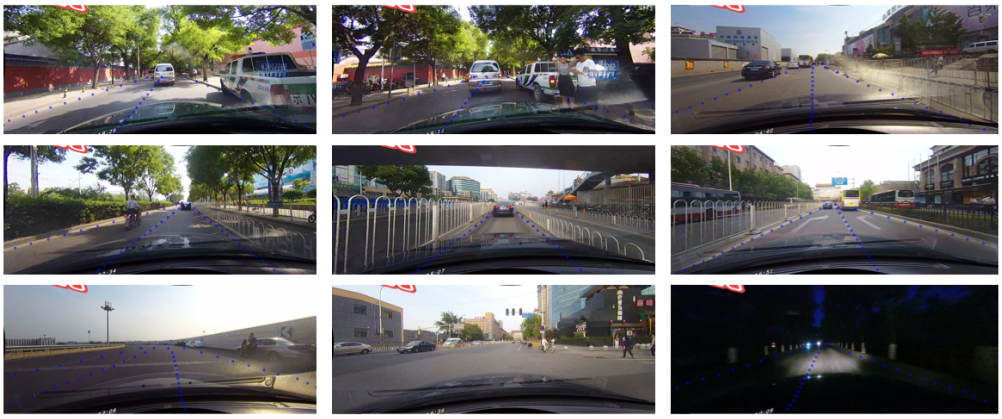

Figure 5: Visualization results of R34-Ours under nine scenarios of CULane, namely, Normal, Crowded, Dazzle, Shadow, NoLine, Arrow, Curve, Cross and Night (from left to right, from top to bottom, best viewed by zooming in). Note that in the Cross situation, only FP is calculated.

XR34-Ours achieves (1) 0.69, 0.93, 3.01, -0.29, 0.76, 0.28, 3.42, 276 and 3.02 gains more than LaneATT, (2) 2.23, 6.26, 10.98, 10.96, 6.75, 4.56, 6.74,936, and 7.64 gains more than SCNN, and (3) 0.93, 6.62, 2.98, 5.86, 3.85, 0.56, 2.54, 842, and 3.94 gains more than RESA on ResNet34. Coupled with the visualization results of R34-Ours in Fig.5, we can safely say that our network has a strong inference ability. It is able to complete the lane instance according to the environment, even in the dark situation (see the ninth image). In terms of running time, R18-Light tops SCNN and RESA by 96 ms and 2 ms, respectively, while keeping satisfying performance. The XR34-Ours is approximately 3× faster than our predecessor SCNN on ResNet34.

TuSimple Results Analysis. The quantitative results on TuSimple in Tab.2 to validate the performance of our network. It can be seen that our

network obtains an accuracy of 96.59 and 96.71 on ResNet-18 and ResNet-34, respectively. The results are very approximate to the highest scores achieved by RESA. Under metric FP, our method gets the best performance of 2.82.

BDD100K Results Analysis. On BDD100k, only some of the semantic segmentation methods can be reported due to the constraints of complex lane forms in Tab.3. This is also one of the most important merits of our AtrousFormer, able to model more complex lanes. The results are showed in Tab3. It can be seen that XR18-Ours surpasses ERFNet-SAD by 22.53 points and 6.79 points in terms of accuracy and IOU metrics, respectively.

4.4. Ablation Studies

Effectiveness of Main Contributions. To validate the effectiveness of the local AtrousFormer enhanced encoder, the global AtrousFormer, and LSGD, we provide detailed ablation studies on CULane in Tab.4. We stipulate ResNet-18+LSGD (without local attention guidance) as the baseline. The segmentation maps and their corresponding existence scores can be obtained after passing the parallel branches of $8\times$ $UpSampling\&Conv2D$ and $8\times$ $Pooling\&MLP$. In Tab.4, the A.F. and XR are used to represent AtrousFormer and locally enhanced ResNet, respectively. We next give some analysis on the adopted techniques.

It can be seen that AtrousFormer is very effective on improving the performance of lane detection. Compared to baseline, equiping global AtrousFormer results in 5.57 points of F1 gains on ResNet-18. The designed decoder LSGD coupled with global AtrousFormer surpasses the global AtrousFormer+LSGD without attention mechanism by 2.02 points in terms of F1 metric, which proves the effectiveness of the local attention guidance. Im-

Table 4: Ablation studies on the test set of CULane against F1, Precision, and Recall metrics.

Methods	F1↑	Precision↑	Recall↑
Baseline	69.60	70.02	69.60
+Global A.F.	75.17	76.66	73.74
+Global A.F.+LSGD	77.19	82.56	72.49
XR18+global A.F.+LSGD	77.63	83.73	72.36
XR34+global A.F.+LSGD	78.08	84.36	72.67

Table 5: The impact of slice size on XR18-Ours

Slice Size	(36, 100)	(18, 20)	(9, 10)	(3, 5)
F1 (R18-Ours)	-	77.63	77.56	56.10

.

proving feature extractor from ResNet-18 to ResNet-34 will generate 0.48 point improvement.

Selectrions of Atrous Experience and Slice Size. To avoid the laboriousness of super-parameter searching, we at first accept the in-layer fusion experience in [18] to bootstrap, namely J = 4, tuning XR18-Ours in a greedy manner. Then we study the impact of slice size of local AtrousFormer on XR18-Ours and report the results in Tab.5. It turns out that when slice size is set to (18, 20), the network gets the best performance. The results confirm that probing lane distribution needs big enough boxes. Then turning up or down J, we find that when J = 4, our network gets the best performance. Along the decreasing of sampling density, the performance of network deteriorates significantly. Related results is listed in Tab.6.

Compared to the Seq2Seq Transformer. In Tab.7, we provide comparison results of the global AtrousFormer and the seq2seq Transformer. All

Table 6: The impact of experience J on XR18-Ours

Experience J	5	4	3	2	1	0
F1 (R18-Ours)	77.61	77.63	77.56	76.42	76.19	74.68

Table 7: Comparison results of the seq2seq Transformer and the global AtrousFormer along the number of heads on ResNet-18+LSGD (The symbol of '-' denotes running out of memory.)

Methods (Heads)	F1↑	Precision↑	Recall↑	K&V Pos. ↓
Seq2seq Trm. (1)	76.19	82.19	71.01	3600
Global A.F. (1)	76.44	82.24	71.41	1224
Seq2seq Trm. (4)	76.37	82.12	71.38	14400
Global A.F. (4)	76.84	82.29	72.05	4896
Seq2seq Trm. (8)	76.42	81.89	71.65	28800
Global A.F. (8)	76.85	82.29	72.08	9792
Seq2seq Trm. (16)	-	-	-	57600
Global A.F. (16)	77.19	82.56	72.49	19584
R.G. A.F. (16)	76.84	82.30	72.07	14400
C.G. A.F. (16)	76.27	82.97	70.57	5184

are conducted on ResNet-18+LSGD by controlling the number of heads. It can be seen that compared to the seq2seq Transformer, in terms of the computation of affinity matrices and F1, the AtrousFormer is more efficient than the seq2seq Transformer given the sparsity of its queried positions. The number comparison is also listed (abbreviated as K&V Pos. in the last column).

Effectiveness of the Two-stage Strategy. We validate the effectiveness of the two-stage strategy in the last two rows in Tab.7. The symbols of R.G. and C.G. represent only row-wise and only column-wise global Atrous-

Formers, respectively. Individually using them will achieve 76.84 and 76.27 points of the F1 score, respectively. The fully global AtrousFormer surpasses the only row-wise one and only column-wise one by 0.35 and 0.98 point, respectively.

5. Conclusion

In this paper, we design AtrousFormer to enhance network inference ability and mine the spatial distribution of lanes. The proposed AtrousFormer mainly has two representative forms, i.e., global AtrousFormer and local AtrousFormer, both of which glean information first by rows and then by columns according to the specified heuristic experience. In this paper, the former is installed on top of the extractor to enhance the global representation of the extracted feature. The latter one has a more compact form, and is inserted into the feature extractor to strengthen extraction. In addition, we design local semantic guided decoder to incorporate the local information into classification stage. Our network achieves impressive performance on CULane, TuSimple and BDD100K. It is able to handle complex situations like occullusion by other vehicles, bad weather conditions, abrasion, terrible illumination intensity, and compleax lane forms.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[2] A. B. Hillel, R. Lerner, D. Levi, G. Raz, Recent progress in road and lane detection: a survey, Machine vision and applications 25 (3) (2014) 727–745.

[3] G. Liu, F. Wörgötter, I. Markelić, Combining statistical hough transform and particle filter for robust lane detection and tracking, in: 2010 IEEE Intelligent Vehicles Symposium, IEEE, 2010, pp. 993–997.

[4] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, H. Chen, A novel lane detection based on geometrical model and gabor filter, in: 2010 IEEE Intelligent Vehicles Symposium, IEEE, 2010, pp. 59–64.

[5] J. Hur, S.-N. Kang, S.-W. Seo, Multi-lane detection in urban driving environments using conditional random fields, in: 2013 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2013, pp. 1297–1302.

[6] Z. Kim, Robust lane detection and tracking in challenging scenarios, IEEE Transactions on intelligent transportation systems 9 (1) (2008) 16–26.

[7] R. Jiang, R. Klette, T. Vaudrey, S. Wang, New lane model and distance transform for lane detection and tracking, in: International Conference on Computer Analysis of Images and Patterns, Springer, 2009, pp. 1044–1052.

[8] A. Borkar, M. Hayes, M. T. Smith, Robust lane detection and tracking with ransac and kalman filter, in: 2009 16th IEEE International Conference on Image Processing (ICIP), IEEE, 2009, pp. 3261–3264.

[9] Y. Jiang, F. Gao, G. Xu, Computer vision-based multiple-lane detection on straight road and in a curve, in: 2010 International Conference on Image Analysis and Signal Processing, IEEE, 2010, pp. 114–117.

[10] H. Tan, Y. Zhou, Y. Zhu, D. Yao, K. Li, A novel curve lane detection based on improved river flow and ransa, in: 17th international ieee conference on intelligent transportation systems (itsc), IEEE, 2014, pp. 133–138.

[11] Z. Chen, Z. Chen, Rbnet: A deep neural network for unified road and road boundary detection, in: International Conference on Neural Information Processing, Springer, 2017, pp. 677–687.

[12] E. Romera, J. M. Alvarez, L. M. Bergasa, R. Arroyo, Erfnet: Efficient residual factorized convnet for real-time semantic segmentation, IEEE Transactions on Intelligent Transportation Systems 19 (1) (2017) 263–272.

[13] T. Liu, Z. Chen, Y. Yang, Z. Wu, H. Li, Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer, in: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2020, pp. 1394–1399.

[14] X. Pan, J. Shi, P. Luo, X. Wang, X. Tang, Spatial as deep: Spatial cnn for traffic scene understanding, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[15] Y. Hou, Z. Ma, C. Liu, C. C. Loy, Learning lightweight lane detection

cnns by self attention distillation, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1013–1021.

[16] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, I. 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, 2017, pp. 1947–1955.

[17] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, M. Hofmann, El-gan: Embedding loss driven generative adversarial networks for lane detection, in: proceedings of the european conference on computer vision (ECCV) Workshops, 2018, pp. 0–0.

[18] T. Zheng, H. Fang, Y. Zhang, W. Tang, Z. Yang, H. Liu, D. Cai, Resa: Recurrent feature-shift aggregator for lane detection, arXiv preprint arXiv:2008.13719.

[19] X. Li, J. Li, X. Hu, J. Yang, Line-cnn: End-to-end traffic line detection with line proposal unit, IEEE Transactions on Intelligent Transportation Systems 21 (1) (2019) 248–258.

[20] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, T. 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, 2021, pp. 294–302.

[21] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, T. Oliveira-Santos, Polylanenet: Lane estimation via deep polynomial

regression, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 6150–6156.

[22] R. Liu, Z. Yuan, T. Liu, Z. Xiong, End-to-end lane shape prediction with transformers, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 3694–3702.

[23] N. Efrat, M. Bluvstein, N. Garnett, D. Levi, S. Oron, B. E. Shlomo, Semi-local 3d lane detection and uncertainty estimation, arXiv preprint arXiv:2003.05257.

[24] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, W. Pedrycz, Key points estimation and point instance segmentation approach for lane detection, IEEE Transactions on Intelligent Transportation Systems.

[25] H. Abualsaud, S. Liu, D. Lu, K. Situ, A. Rangesh, M. M. Trivedi, Laneaf: Robust multi-lane detection with affinity fields, arXiv preprint arXiv:2103.12040.

[26] H. Xu, S. Wang, X. Cai, W. Zhang, X. Liang, Z. 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, Springer, 2020, pp. 689–704.

[27] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, T. Darrell, Bdd100k: A diverse driving dataset for heterogeneous multitask learning.

[28] M. Yang, K. Yu, C. Zhang, Z. Li, K. Yang, Denseaspp for semantic segmentation in street scenes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3684–3692.

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

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929.

[31] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, arXiv preprint arXiv:2103.14030.

[32] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, L. Shao, Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, arXiv preprint arXiv:2102.12122.

[33] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, Q. Ye, Conformer: Local features coupling global representations for visual recognition, arXiv preprint arXiv:2105.03889.

[34] H. Law, J. Deng, Cornernet: Detecting objects as paired keypoints, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 734–750.

[35] J. 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, 2019, pp. 11582–11591.

[36] R. Fan, X. Wang, Q. Hou, H. Liu, T.-J. Mu, Spinnet: Spinning convolutional network for lane boundary detection, Computational Visual Media 5 (4) (2019) 417–428.

[37] Z. Qin, H. Wang, X. 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, Springer, 2020, pp. 276–291.

[38] S. Yoo, H. S. Lee, H. Myeong, S. Yun, H. Park, J. Cho, D. H. 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, 2020, pp. 1006–1007.

[39] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, IEEE.

[40] Y. Hou, Z. Ma, C. Liu, T.-W. Hui, C. C. Loy, Inter-region affinity distillation for road marking segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12486–12495.