Laneformer: Object-aware Row-Column Transformers for Lane Detection

Jianhua Han¹, Xiajun Deng², Xinyue Cai¹, Zhen Yang¹, Hang Xu¹, Chunjing Xu¹, Xiaodan Liang²∗

¹Huawei Noah’s Ark Lab, ²Shenzhen Campus of Sun Yat-sen University

Abstract

We present Laneformer, a conceptually simple yet powerful transformer-based architecture tailored for lane detection that is a long-standing research topic for visual perception in autonomous driving. The dominant paradigms rely on purely CNN-based architectures which often fail in incorporating relations of long-range lane points and global contexts induced by surrounding objects (e.g., pedestrians, vehicles). Inspired by recent advances of the transformer encoder-decoder architecture in various vision tasks, we move forwards to design a new end-to-end Laneformer architecture that revolutionizes the conventional transformers into better capturing the shape and semantic characteristics of lanes, with minimal overhead in latency. First, coupling with deformable pixel-wise self-attention in the encoder, Laneformer presents two new row and column self-attention operations to efficiently mine point context along with the lane shapes. Second, motivated by the appearing objects would affect the decision of predicting lane segments, Laneformer further includes the detected object instances as extra inputs of multi-head attention blocks in the encoder and decoder to facilitate the lane point detection by sensing semantic contexts. Specifically, the bounding box locations of objects are added into Key module to provide interaction with each pixel and query while the ROI-aligned features are inserted into Value module. Extensive experiments demonstrate our Laneformer achieves state-of-the-art performances on CULane benchmark, in terms of 77.1% F1 score. We hope our simple and effective Laneformer will serve as a strong baseline for future research in self-attention models for lane detection.

Introduction

Lane detection has been a long-standing task for visual perception in autonomous driving scenarios, targeting at precisely distinguishing lane segments from complicated road scenes (Hou et al. 2019; Wang, Shen, and Teoh 2000; Narote et al. 2018). It plays a crucial role towards a safe and reliable auto-driving system widely used in intelligent cars to assist drivers with modern technologies such as adaptive cruise control, lane departure warning, and traffic understanding.

The most state-of-the-art lane detection methods (Lee et al. 2021; Xu et al. 2020; Chen, Liu, and Lian 2019) take advantage of CNN architectures and exceed the traditional methods (Niu et al. 2016; Narote et al. 2018; Wang, Shen, and Teoh 2000) by a large margin. However, the existing CNN-based methods usually need complicated post-processing procedures like non-maximal suppression or clustering. In addition, the fixed receptive field of CNN architecture limits the ability to incorporate relations for long-range lane points, making them hard to capture well the characteristics of lanes since the shapes are conceptually long and thin. Several attention-based lane detection models (Lee et al. 2021; Tabelini et al. 2020b; Liu et al. 2021) have been also proposed to capture the long-range information. Nevertheless, the fixed attention routines can not adap-
rettively fit the shape characteristic of lanes. Besides, complicated road scenes including different light, weather conditions and occlusions of surrounding objects further require the model with a stronger perception of global contexts. In addition to the point-wise context information, it is reasonable to assume that objects on the road (e.g., pedestrians, vehicles) have some implicit relations with surrounding lanes. As the key sub-module (e.g., person-vehicle detection) of autonomous driving systems usually co-exist with lane detection, the sub-module outputs may promote the performance of lane detection and improves the system safety. However, none of the existing methods has considered incorporating semantics induced by detection results from a single and unified network view.

On the other hand, Transformer [Vaswani et al. 2017], a kind of encoder-decoder architecture, has shown surprisingly promising ability in dealing with tasks that require to capture global relations in nature language processing [Devlin et al. 2018; Radford et al. 2018; Young et al. 2018] and vision tasks such as image classification [Dosovitskiy et al. 2020], object detection [Carion et al. 2020; Zhu et al. 2020] and image segmentation [Zheng et al. 2020; Wang et al. 2020]. Particularly, [Liu et al. 2021] proposed an LSTR model to predict lanes as polynomial functions using a transformer. Despite its benefit in providing rich global contexts in predicting lanes, the definition that regards lanes as polynomial functions has many drawbacks: 1) LSTR needs to formulate camera intrinsic and extrinsic parameters, which hinders the model from transferring to other datasets or train with combined datasets; 2) LSTR still lacks of explicitly modeling global semantic contexts into facilitating lane detection.

To tackle the above-mentioned issues, we move forwards to design a new Laneformer architecture to better capture the shape characteristics and global semantic contexts of lanes using transformers (shown in Figure 1). Our Laneformer defines lanes as a series of points. Then, the lane-specific row-column attentions are proposed to efficiently mine point context along with the lane shapes. Concretely, we take each feature row as a token and perform row-to-row self-attention and do the same for each feature column to perform column-to-column self-attention. Moreover, motivated by the appearing objects that would affect lanes’ predictions, Laneformer further includes the detected object instances as auxiliary inputs of multi-head attention blocks in encoder and decoder to facilitate the lane point detection by sensing semantic contexts. To be specific, the bounding box locations of objects are added into the Key module to provide interaction with each pixel and query. At the same time, the ROI-aligned features are inserted into the Value module to provide detection information. Considering that bounding box locations and ROI-aligned features contain limited information about object instances, we use confident scores and predicted categories of the detected outputs to further improve the model’s performance. Bipartite matching is adopted to ensure one-to-one alignment between predictions and ground truths, which makes the Laneformer architecture eliminate additional post-processing procedures.

Extensive experiments conducted on CULane [Pan et al. 2018] and TuSimple [TuSimple 2017] benchmarks show that our Laneformer as an early transformer attempt on lane detection, achieves state-of-the-art performance on CULane with 77.1% F1 score and superior 96.8% accuracy on TuSimple, with about 50 FPS(frames-per-second) inference speed on V100. Besides, visualization of the learned attention map in transformer demonstrates that our Laneformer can incorporate relations of long-range lane points and global contexts induced by surrounding objects.

To summarize, the main contributions can be listed:

- We design a new Laneformer architecture specified for lane detection, which accommodates the conventional transformers into capturing well the shape characteristics and semantic contexts of lanes.
- The lane-specific row-column self-attentions is proposed to mine point context along with the lane shapes with the prior that lanes are long and thin.
- The object instances by detection that usually co-exist in autonomous driving system are used as auxiliary inputs to sense object-wise semantic contexts.
- Experiments show that Laneformer achieves new state-of-the-art performance, in terms of 77.1% F1 score on CULane and superior 96.8% accuracy on TuSimple.

**Related work**

**Attention-free methods.** Before the advent of deep learning, traditional lane detection methods are usually based on hand-crafted low-level features [Chiu and Lin 2005; Lee and Cho 2009; Gonzalez and Ozguner 2000]. CNN architecture has then been adopted to extract advanced features in an end-to-end manner. Most lane detection methods follow pixel-level segmentation-based approach [Pizzati et al. 2019; Hou 2019; Mamidala et al. 2019; Zou et al. 2019]. These approaches typically generate segmentation results with an encoder-decoder structure and then post-processing them via curve fitting and clustering. However, pixel-wise lane prediction methods usually require more computation and are also limited to the pre-defined number of lanes. On the other hand, several works [Chen, Liu, and Lian 2019; Li et al. 2019; Xu et al. 2020] follow traditional proposal-based diagrams by generating multiple point anchors and then predicting the relative distance between each lane point and the anchor point. However, these existing CNN-based methods usually need complicated post-processing procedures like non-maximal suppression or clustering. Besides, the fixed receptive field of CNN architecture limits the ability to incorporate relations for long-range lane points. Therefore, our Laneformer utilizes a transformer to capture context information and bipartite matching to eliminate additional post-processing procedures.

**Attention-based methods.** Several attention-based lane detection models [Lee et al. 2021; Tabelini et al. 2020b; Liu et al. 2021] have been proposed to capture the long-range information. [Lee et al. 2021] propose a self-attention mechanism to predict the lanes’ confidence along with the vertical and horizontal directions in an image. [Tabelini et al. 2020b] proposes a novel anchor-based attention mechanism that aggregates global information named LaneATT. However, the...
fixed attention routines can not adaptively fit the shape characteristic of lanes. Based on PolyLaneNet(Tabelini et al. 2020a), LSTR(Liu et al. 2021) is proposed to output polynomial parameters of a lane shape function by using a network built with a transformer to learn richer structures and context. Unlike LSTR, which assumes all lanes are parallel on the road and formulates it as a polynomial shape prediction problem, our Laneformer directly outputs each lane’s points to adapt to more complex lane detection scenarios. Besides, detected object instances are further included as auxiliary inputs of multi-head attention blocks in Laneformer to facilitate the lane point detection by sensing semantic contexts.

**Laneformer**

**Lane Representation**

Similar to (Chen, Liu, and Lian 2019), we define lanes as a series of 2D-points that can adapt to all kinds of lanes. Specifically, we formulate a lane as \( l = (X, s, e) \), where \( X \) stands for corresponding x-coordinate set for 72 equally-spaced y-coordinates and \( s, e \) denote for the start y-coordinate and end y-coordinate of the lane.

**Architecture**

The overall architecture of Laneformer is demonstrated in Figure 2. It mainly consists of four modules: a CNN backbone to extract basic features, a detection processing module to handle outputs from person-vehicle detection module, a specially designed encoder and a decoder for lane detection.

Given an RGB image as input, our model first extracts backbone features with a ResNet (He et al. 2016) backbone. We further get row features and column features by collapsing the column dimension and row dimension of backbone features, respectively. At the same time, detected bounding boxes and their predicted scores and categories from the input image are acquired through a trained person-vehicle detection module. We use the bounding box locations to crop ROI-aligned features from backbone features mentioned above, followed by a 1-layer perceptron with ReLU activation to transform the ROI-aligned features to one-dimensional embeddings. Similarly, another 1-layer perceptron is applied on the detected bounding boxes to get one-dimensional bounding box embeddings. In the encoder, row attention and column attention are performed on row features and column features respectively. Meanwhile, backbone features will perform self-attention and pixel-to-Bbox attention to adapt to more complex lane detection scenarios. The outputs of row-column attention and pixel-to-Bbox attention are added up as the memory input for the following decoder. In the decoder, learnable queries first perform self-attention to get query features. Then the query features together with the input memory will perform cross attention. Meanwhile, with the bounding box embeddings and ROI embeddings, query-to-Bbox attention can be applied. Finally, outputs of the cross attention and query-to-Bbox attention are added up, and several feed-forward layers are utilized to predict lane points.

**Detection Processing**

After the acquisition of detected bounding boxes, predicted scores and predicted categories from a trained detector based on common Faster-RCNN(Ren et al. 2015) architecture, we propose a simple detection processing module to process them in order to better utilize the detection information. First, we crop ROI-aligned features from backbone features \( H_f \in R^{h \times w \times d} \) for the bounding boxes of the detected objects, where \( h, w \) and \( d \) are the spatial sizes and the corresponding dimension of backbone features. Since the features we used are down-sampled, we need to rescale the bounding box locations by a specific ratio to crop out correct fea-
At the pre-training stage, we focus on the basic attention module and detection attention. Considering that objects with higher confidence scores may supply more robust information while objects with lower scores may be noisy, we also use predicted scores as weight coefficients to multiply the ROI-aligned features and get weighted ROI-aligned features for each object. After that, we pass through weighted ROI-aligned features into a 1-layer perceptron with output channel $d$. That, we pass through weighted ROI-aligned features into the encoder or query feature $f$. Similarly, with the dimension-reduced column features $H_c \in R^{1 \times w \times d'}$, and its position embeddings, self-attention are performed between columns. Finally, the outputs of row and column attention are added up as the memory input for decoder. Details are shown in Figure 3(a).

**Row-Column Attention**

In the encoder of Laneformer, besides the traditional attention, we further propose the new row-column attention to efficiently mine point context along with the lane shapes. The row-column attention consists of row attention that captures the relations between rows and column attention that mine the relations between columns. As we all know, the traffic lane is a kind of object with unique shape characteristics. From the vertical view, lanes are long and thin, which means pixels in the same lane will not be far from each other between adjacent rows. From the horizontal view, since different columns may cross different lanes, the knowledge sharing of these columns may capture different lane features to construct better representations.

Given row features $H_r \in R^{h_1 \times 1 \times wd}$, where $h_1$, $w$, and $d$ are the spatial sizes and the corresponding dimension of backbone features, we pass it through a linear transformation to reduce the last dimension to $d'$, which we denoted as $H'_r \in R^{h_1 \times 1 \times d'}$. Sinusoidal embeddings $E_h$ is calculated according to the absolute positions to supply location information. Self-attention operations are applied with the inputs of $H'_r$ and $E_p$. Similarly, with the dimension-reduced column features $H_c \in R^{1 \times w \times d'}$ and its position embeddings, self-attention are performed between columns. Finally, the outputs of row and column attention are added up as the memory input for decoder. Details are shown in Figure 3(a).

**Detection Attention**

Apart from row-column attention, in both the encoder and decoder of Laneformer, we propose detection attention to mine the valuable information from detected surrounding objects. It is straightforward that lanes and objects on the road (e.g., pedestrians, vehicles) have some implicit relations with each other. For example, lanes are more likely to appear next to vehicles, while pedestrians are not supposed to walk between lanes. Besides, object detection module such as person-vehicle detection is always co-existed with lane detection in an auto-driving system, thus detected results can be acquired easily. The detection attention module consists of two parts: 1) a Pixel-to-Bbox attention module in encoder to excavate the relevance between each pixel token of feature map and each detected objects; 2) a Query-to-Bbox attention module in decoder to find out which object should be paid more attention in order to help predict corresponding lanes. Details are illustrated in Figure 3(b).

**Pixel-to-Bbox Attention** In Laneformer encoder, we propose pixel-to-Bbox attention. Pixel-to-Bbox attention is designed for digging out relations between feature pixels and detected objects. In pixel-to-Bbox attention, each pixel in dimension-reduced backbone features $H'_r$ is considered as a query token. Detection information including bounding box embeddings $Z_b$ and ROI embeddings $Z_r$ make up the Key module and Value module, respectively. The pixel-to-Bbox attention is defined as follows:

$$O_{p2b} = \text{softmax} \left( \frac{H'_r Z_b^T}{\sqrt{d'}} \right) \cdot Z_r$$

(1)

Pixel-to-Bbox attention forces the model to learn which detected object a pixel should pay attention to so that the model can catch helpful context features. The output of pixel-to-Bbox attention $O_{p2b}$ are added up with the above row-column attention output and act as the memory input for the decoder.

**Query-to-Bbox Attention** In Laneformer decoder, we propose query-to-Bbox attention. Query-to-Bbox attention on the other hand designed to mine relations between queries and detected objects. Query-to-Bbox attention is similar to Pixel-to-Bbox attention, while in here, queries are the
learned embeddings $Q$ with the size of $N \times d'$, where $N$ is the number of learned embeddings. Bounding box embeddings $Z_b$ and ROI embeddings $Z_r$ still act as Key module and Value module here. Similarly, the query-to-Bbox attention is defined as follows:

$$O_{q2b} = \text{softmax} \left( \frac{QZ_b^T}{\sqrt{d'}} \right) \cdot Z_r$$

(2)

This attention enables each query to focus on the instances near the lane it needs to predict. The output of query-to-Bbox attention $O_{q2b}$ will be added up with the traditional cross-attention output, followed by several feed-forward features to get predicted lanes.

**Loss Construction**

**Bipartite matching.** After obtaining features from above-mentioned modules, Laneformer predicts a number of $N$ lane set according to its number of queries, where $N$ is set to be significantly larger than the maximum number of lanes in the dataset. Therefore we need to pad the ground truth with non-lane to be a set of $N$ objects first, denoted as $G = \{ g_n | g_n = (c_n, l_n) \}_{n=1}^N$, where $c_n \in \{0, 1\}$, 0 represents non-lane and 1 represents lane. Given predicted outputs as $P = \{ p_n | p_n = (\hat{p}_n, \hat{l}_n) \}_{n=1}^N$, where \( \hat{p}_n(c_n) \) stands for the probability score that \( \hat{l}_n \) belongs to the specific category \( c_n \). We search for a permutation of $N$ elements matching index $\delta$ to minimize the pair-wise distance function $D$ between ground-truth lane $g_n$ and predicted lane $p_{\delta(n)}$:

$$\hat{\delta} = \arg \min_\delta \sum_{n=1}^N D(g_n, p_{\delta(n)}).$$

(3)

The difference function $D$ is defined as following:

$$D(g_n, p_{\delta(n)}) = -\omega_1 \hat{p}_{\delta(n)}(c_n) + \mathbb{I}(c_n = 1) L_{loc}(l_n, \hat{l}_{\delta(n)})$$

(4)

Where $\mathbb{I}(\ast)$ denotes an indicator function and $\omega_1$ stands for the coefficient of classification term. $L_{loc}$ is defined as follows:

$$L_{loc}(l_n, \hat{l}_{\delta(n)}) = \omega_2 L_1(X_n, \hat{X}_{\delta(n)}) + \omega_3 L_1(s_n, \hat{s}_{\delta(n)}) + \omega_4 L_1(e_n, \hat{e}_{\delta(n)})$$

(5)

where $L_1$ denotes the mean absolute error and $\omega_2, \omega_3, \omega_4$ indicate for the coefficients for point, start position and end position respectively. Bipartite matching is adopted to ensure one-to-one alignment between predictions and ground truths, making the Laneformer an end-to-end architecture by eliminating additional post-processing procedures.

**Total Loss** The total loss of our model is calculated with matching index $\delta$ gained from bipartite matching, consisting of negative log-likelihood loss for classification prediction and $L1$-based location loss:

$$L_{total} = \sum_{n=1}^N \omega_1 \log \hat{p}_{\delta(n)}(c_n) + \mathbb{I}(c_n = 1) L_{loc}(l_n, \hat{l}_{\delta(n)})$$

(6)

Where $L_{loc}$ is calculated the same with Eq. (5) and $(n, \delta(n))$ is the optimal pair indexes that minimize Eq. (3). $\omega_1, \omega_2, \omega_3, \omega_4$ also adjust the effect of the loss terms and are set as same values with Eq. (4) and Eq. (5).

**Experiment**

**Datasets and Evaluation Metrics**

We conduct experiments on the two most popular lane detection benchmarks. **CULane** (Pan et al. 2018) is a large-scale traffic lane detection dataset that is collected by in-vehicle cameras in Beijing, China. It consists of 88,880 training images, 9675 validation images, and 34,680 test images. The test split is further divided into normal and 8 challenging categories. **TuSimple** (TuSimple 2017) is an autonomous driving dataset which specifically focuses on real highway scenarios, including 3626 images for training set and 2782 images for the test set.

**Evaluation Metrics** We use F1 score (abbreviated as F1 in the following section) to measure the model performance on CULane and 640 x 360 for TuSimple. Data augmentations are applied on the raw image, consisting of horizontal flipping, a random choice from color-shifting operations (e.g., gaussian blur, linear contrast) and position-shifting operations (e.g., cropping, rotate). Most of our experiments use ResNet50 as the backbone. We follow the setting of (Zhu et al. 2020) and utilize a deformable transformer as the plain transformer. The bipartite matching and loss term coefficients $\omega_1, \omega_2, \omega_3$ and $\omega_4$ are set as 2, 10, 10, 10, respectively. Both the number of encoder and decoder layers is set to 1. Moreover, we adopt 25 as the number of queries $N$ and 10 as the number of used detected bounding boxes $M$. Eight V100s are used to train the model and the batch size is set to be 64. The learning rate is set to 1e-4 for the backbone and 1e-5 for the transformer. We train 100 epochs on CULane and drop the learning rate by ten at 80 epoch. On Tusimple, the total number of training iterations is set to 28k and the learning rate drops at 22k iteration. During inference, the single scale test is adopted with the score threshold set to 0.8. The trained detector used to obtain person-vehicle bounding boxes is based on common Faster-RCNN (Ren et al. 2015) architecture with ResNet-50 backbone and trained 12 epochs on BDD100K dataset (Yu et al. 2018) with 70k images.

**Main Results**

**CULane.** Table 1 shows the Laneformer’s performance on CULane test set. Our Laneformer achieves state-of-the-art results on F1 of the total test split. In addition, our Laneformer with only ResNet-50 backbone even surpasses the results of LaneATT (Labeleni et al. 2020b) with a larger ResNet-122 backbone. Additionally, Laneformer outperforms all other lane detection models on some challenging splits such as “Night”, “Dazzle light” and “Cross”. Among the above three splits, Laneformer significantly improves the performance of “Cross” category, which achieves an extremely low FP (False Positive) number 19. We observe
Table 1: Comparison of F1-measure(%) and MACs(multiply–accumulate operations) on CULane testing set, where Laneformer* denotes for Laneformer without detection attention module. Our Laneformer achieves state-of-the-art performance.

| Models             | ACC    | F1(%)  | Precision(%) | Recall(%) | FPS     | Params. (M) |
|--------------------|--------|--------|--------------|-----------|---------|-------------|
| SCNN               | 96.53  | 6.17   | 1.80         |           |         |             |
| LSTR               | 96.18  | 2.91   | 3.38         |           |         |             |
| PolyLaneNet        | 96.34  | 4.67   | 5.18         |           |         |             |
| LaneATT (ResNet-18)| 95.76  | 3.53   | 3.01         |           |         |             |
| LaneATT (ResNet-34)| 95.63  | 3.53   | 2.92         |           |         |             |
| LaneATT (ResNet-50)| 95.10  | 4.64   | 2.54         |           |         |             |

Latency Comparison. In inference, Laneformer with ResNet-50 backbone achieves 53 and 48 FPS on one V100 for CULane and Tusimple benchmark. For latency comparison of different components in Laneformer, we conduct experiments on CULane testing split and illustrate the result in Table 3. After adding row-column attention and detection attention, there is only a 4.9%, 8.1% increment on inference FPS due to the efficient matrix multiplication.

Visualization. We visualize several attention maps in the transformer to find out the area that detection attentions and row-column attentions focus on, where brighter color denotes for more significant attention value. Shown in Figure 4(a), either the point or query pays more attention to the detected instances besides lanes it responsible for, especially when those instances are in occlusion with part of the lane. Moreover, observation can be made in Figure 4(b) that row attention mainly considers nearby rows, while the column attention focuses on the nearby representative column of each lane. These results demonstrate our assumption that the implicit relationship of traffic scenes can be obtained from proposed detection attention and row-column attention.

Ablation Study

Different components. As we can see in Table 3 without detection attention and row-column attention, our baseline (plain transformer) obtains 75.45% on F1. The adding of row-column attention improves the performance to 76.04%. What’s more, simply introducing detected objects information can improve the performance of our model, but a little more extra information such as scores or categories will make it better. We can observe that making full use of detected objects with their scores and categories raises the F1 to 77.06%, which is the state-of-the-art results on CULane.

Score threshold of the bounding box. To find out the influences of detected objects with different confidence, we conduct a series of experiments with different score thresholds on filtering detected bounding boxes. Table 4 shows that using detection outputs from different score thresholds has a slight impact on our results. Especially, results between threshold 0.6 to 0.8 are robust and the differences can be nearly neglect. When the threshold is lower than 0.6, there may be some noise in detected bounding boxes and therefore
Table 4: Quantitative evaluation of different detection bounding box input settings on CULane testing split.

| Score threshold | F1(%) | Precision(%) | Recall(%) |
|-----------------|-------|--------------|-----------|
| 0.4             | 76.34 | 85.41        | 69.01     |
| 0.5             | 76.14 | 84.52        | 69.27     |
| 0.6             | 77.06 | 84.05        | 71.14     |
| 0.7             | 76.71 | 84.53        | 70.21     |
| 0.8             | 76.85 | 84.51        | 70.46     |
| 0.9             | 76.37 | 85.54        | 68.98     |
| random         | 75.99 | 84.50        | 69.03     |

| Number of Bbox | F1(%) | Precision(%) | Recall(%) |
|----------------|-------|--------------|-----------|
| 5              | 76.90 | 84.30        | 70.69     |
| 10             | 77.06 | 84.05        | 71.14     |
| 20             | 76.51 | 84.34        | 70.01     |

| Different categories | F1(%) | Precision(%) | Recall(%) |
|----------------------|-------|--------------|-----------|
| none                 | 76.04 | 82.92        | 70.22     |
| person               | 76.40 | 85.17        | 69.27     |
| vehicle              | 76.79 | 84.44        | 70.41     |
| all                  | 77.06 | 84.05        | 71.14     |

Figure 4: Visualization of proposed attentions on Tusimple dataset, where brighter color denotes more significant attention value. (a) shows the detection attention map for different points and queries in the transformer. We can observe either the pixel or the query focus on the detection bounding boxes near the corresponding lane. (b) shows that the row attention mainly considers the nearby rows, while the column attention focuses on the nearby representative column of each lane.

Table 4: Quantitative evaluation of different detection bounding box input settings on CULane testing split.

- **Number of bounding box.** Apart from thresholds, we further explore the impact of using a different number of detected objects in Laneformer. The results in Table 4 show that too many detected objects lead to a performance drop, and our model reaches the best result under the setting of 10 bounding boxes. We speculate that the best setting of 10 bounding boxes is due to the average number of detected objects in each image under the bounding box threshold of 0.6, which is 9.84. So if we set a number much larger than 9.84, for example, 20 in our experiment, then too many useless padded boxes will be used, which may hurt the model’s performance. On the other hand, if we use too few bounding boxes, the information of detected objects is not entirely utilized to reach the best performance.

- **Different categories.** Vehicles and persons have different relations with lanes. To be specific, vehicles are on the road, beside the lanes, while persons usually far from lanes. So we also explore the impact of extra information with different categories of bounding boxes. Results in Table 4 show that both the adding of vehicles and persons can improve the model performance, and the result with vehicles is better than the one with persons. We suppose that vehicles share a more close relation with lanes in the perspective of locations. Experiments show that the model with all of the two categories reaches the best results.

**Conclusions**

In this work, we propose Laneformer, a conceptually simple yet powerful transformer-based architecture tailored for lane detection. Equipped with row and column self-attention module and semantic contexts provided by additional detected object instances, Laneformer achieves the state-of-the-art performance, in terms of 77.1% F1 score on CULane and superior 96.8% Accuracy on TuSimple benchmark. Besides, visualization of the learned attention map in transformer demonstrates that our Laneformer can incorporate relations of long-range lane points and global contexts induced by surrounding objects.
References

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In European Conference on Computer Vision, 213–229. Springer.

Chen, Z.; Liu, Q.; and Lian, C. 2019. PointLaneNet: Efficient end-to-end CNNs for Accurate Real-Time Lane Detection. In IV, 2563–2568. IEEE.

Chiu, K.-Y.; and Lin, S.-F. 2005. Lane detection using color-based segmentation. In IV, 706–711. IEEE.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Gonzalez, J. P.; and Ozguner, U. 2000. Lane detection using histogram-based segmentation and decision trees. In Proceedings (Cat. No. 00TH8493).

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In CVPR.

Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS.

Tabelini, L.; Berriel, R.; Paixão, T. M.; Badue, C.; De Souza, A. F.; and Oliveira-Santos, T. 2020a. PolyLaneNet: Lane estimation via deep polynomial regression. arXiv preprint arXiv:2004.10924.

Tabelini, L.; Berriel, R.; Paixão, T. M.; Badue, C.; De Souza, A. F.; and Oliveira-Santos, T. 2020b. Keep your Eyes on the Lane: Attention-guided Lane Detection. arXiv preprint arXiv:2010.12035.

TuSimple. 2017. Tusimple lane detection challenge. In CVPR Workshops.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In NIPS.

Wang, H.; Zhu, Y.; Adam, H.; Yuille, A.; and Chen, L.-C. 2020. MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers. arXiv preprint arXiv:2012.00759.

Xu, H.; Wang, S.; Cai, X.; Zhang, W.; Liang, X.; and Li, Z. 2020. CurveLane-NAS: Unifying Lane-Sensitive Architecture Search and Adaptive Point Blending. arXiv preprint arXiv:2007.12147.

Yu, F.; Xian, W.; Chen, Y.; Liu, F.; Liao, M.; Madhavan, V.; and Darrell, T. 2018. BDD100K: A diverse driving video database with scalable annotation tooling. arXiv preprint arXiv:1805.04687.

Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P. H.; et al. 2020. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. arXiv preprint arXiv:2012.15840.

Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; and Dai, J. 2020. Deformable DETR: Deformable Transformers for End-to-End Object Detection. arXiv preprint arXiv:2010.04159.

Zou, Q.; Jiang, H.; Dai, Q.; Yue, Y.; Chen, L.; and Wang, Q. 2019. Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks. CoRR, abs/1903.02193.

Pan, X.; Shi, J.; Luo, P.; Wang, X.; and Tang, X. 2018. Spatial as deep: Spatial cnn for traffic scene understanding. In AAAI.

Pizzato, F.; Allodi, M.; Barrera, A.; and García, F. 2019. Lane Detection and Classification using Cascaded CNNs. CoRR, abs/1907.01294.

Wang, Y.; Shen, D.; and Teoh, E. K. 2000. Lane detection using spline model. Pattern Recognition, 34(5): 941–947.

Zou, Q.; Jiang, H.; Dai, Q.; Yue, Y.; Chen, L.; and Wang, Q. 2019. Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks. CoRR, abs/1903.02193.
Appendix

We conduct more ablation studies to confirm the effectiveness of Laneformer. In the following section, we will show the results of these experiments based on ResNet-50.

Layers of Encoder and Decoder

It can be seen in Table 5, the different number of encoder layers and decoder layers have a slight difference in performances, which proves that our Laneformer is robust. We can observe that the increment of encoder layers and decoder layers can improve the performance on Recall metric.

Table 5: Experiments on different numbers of encoder and decoder layers on CULane.

| Encoder | Decoder | F1(%) | Precision(%) | Recall(%) |
|---------|---------|-------|--------------|-----------|
| 1       | 1       | 77.06 | 84.05        | 71.14     |
| 2       | 2       | 76.48 | 82.49        | 71.28     |
| 3       | 3       | 76.91 | 82.59        | 71.97     |
| 1       | 2       | 76.90 | 82.92        | 71.70     |
| 1       | 3       | 76.71 | 82.95        | 71.35     |
| 2       | 1       | 76.07 | 80.13        | 72.39     |
| 3       | 1       | 76.13 | 81.87        | 71.14     |

Threshold and Bounding Box Number

As we have mentioned in our paper, the number of bounding boxes and the score threshold of bounding boxes have a relationship that they will influence the choice of each other. We do another two experiments on threshold 0.4 and 0.9 to verify this influence. First, we count the average objects number in each image under threshold 0.4 and 0.9, which is 11 objects and 7 objects, respectively. Then, we conduct experiments in two settings. Concretely, we use 11 bounding boxes for 0.4 score threshold and 7 bounding boxes for 0.9 score threshold. As we can see in Table 6, using the proper number of bounding boxes under corresponding threshold can clearly improve the performances.

Table 6: Experiments on different score threshold and bounding box number on CULane.

| Threshold | Number | F1(%) | Precision(%) | Recall(%) |
|-----------|--------|-------|--------------|-----------|
| 0.4       | 10     | 76.34 | 85.41        | 69.01     |
|           | 11     | 76.77 | 84.88        | 70.08     |
| 0.9       | 10     | 76.37 | 85.54        | 68.98     |
|           | 7      | 76.46 | 84.61        | 69.74     |

Different Detectors

To find out the influence of person-vehicle detectors with different accuracy, we also conduct experiments on different Faster-RCNNs trained on 10k, 30k, 70k images chosen from BDD100K dataset. Random bounding box setting is added for comparison. Results in Table 7 demonstrate that our Laneformer can work well with detectors of different accuracy. Moreover, with the improvement of detector’s accuracy, the performance of Laneformer also improves.

Table 7: Experiments on different detectors, ROI sizes and different layers on CULane.

| Different Detectors | F1(%) | Precision(%) | Recall(%) |
|---------------------|-------|--------------|-----------|
| random              | 75.99 | 84.50        | 69.03     |
| 10k                 | 76.71 | 85.36        | 69.65     |
| 30k                 | 76.96 | 85.10        | 70.25     |
| 70k                 | 77.06 | 84.05        | 71.14     |

| ROI size            | F1(%) | Precision(%) | Recall(%) |
|---------------------|-------|--------------|-----------|
| 3 × 3               | 77.06 | 84.05        | 71.14     |
| 5 × 5               | 76.06 | 83.80        | 69.62     |
| 7 × 7               | 76.05 | 80.72        | 71.88     |

| Different Layers    | F1(%) | Precision(%) | Recall(%) |
|---------------------|-------|--------------|-----------|
| C3                  | 76.47 | 85.76        | 69.00     |
| C4                  | 77.06 | 84.05        | 71.14     |
| C5                  | 75.94 | 80.75        | 71.67     |

ROI Sizes

We can observe from Table 7 that the bigger the extracted ROI features, the lower of F1 score. We speculate that we extract ROI features from C4 layer of backbone features, which is of size 52 × 19 that is very small, so ROI features of size 3 × 3 is enough. If the size goes too big, the extracted feature needs to be interpolated to the specified size, which may introduce fuzzy information.

ROI Features from Different Layers

Our Laneformer extracts multi-stage features from the backbone, so we have choices to extract ROI features from backbone feature of different layers. We try for C3, C4 and C5 layers. Table 7 shows that we get our best performance on ROI features extracted from C4 layer. While features extracted from C5 layer show a significant performance drop due to the feature size is too small that it loss too much information.
(a) Lane detection results on TuSimple dataset.

(b) Lane detection results on CULane dataset.

Figure 5: Visualization of Laneformer results on TuSimple and CULane dataset.