CTLane: An End to End Lane Detector by CNN-Transformer and Fusion Decoder

Guoqiang Zhu[1†], Mian Zhou[1*†], Yanbing Xue[1], Guangping Xu[1] and Zhouming Qi[1]

[1]School of Computer Science and Engineering, Tianjin University of Technology, 391 Binshuixidao Road, Tianjin, 300384, Tianjin, China.

*Corresponding author(s). E-mail(s): zhoumian@tjut.edu.cn;
Contributing authors: vase@stud.tjut.edu.cn; xueyb0718@tjut.edu.cn; xugp@tjut.edu.cn; qimingtiger@stud.tjut.edu.cn;
†These authors contributed equally to this work.

Abstract

In advanced driving assisted system or autonomous driving system, lane detection is an important part for intelligent vehicles in driving and safety. Deep learning based lane detection methods have provide high precise pixel category prediction, but fail to interpret lanes as whole when interference occurs. Hence, in our method, we have designed two components: CNN-Transformer and Fusion Decoder. The former is to extract the whole semantics and to accelerate the convergence speed. The latter aggregates high-level semantics and low-level local feature. We tested the method on several lane datasets and achieved superior results, with the best results on the BDD100K dataset. Our method has solved the problem that the lane cannot be correctly and completely recognized due to the interference caused by dark night, shadows, strong light, etc. The code has been made available at: https://github.com/squirtlecc/CNNTransformer.

Keywords: Lane Detection, Transformer, Convolution, Deep Learning

1 Introduction

Lane detection, as an important perception part of current advanced driving assisted system or autonomous driving systems, plays an important role for keeping vehicles safe and direct vehicles driving forward. Nowadays, with the surge of deep learning, many standard methods [1, 2, 3] consider Lane Detection as a segmentation task and utilize an end-to-end neural network framework. As segmentation takes, the model is trying to focus every pixel and predict the category of each pixel, which leads to the model fails to take the lane line as a whole and lose the semantics of lanes in feature maps. The situation becomes worse when the illumination is not ideal, e.g., shadow cast on lanes, vehicles blocking lanes, or night driving.

Currently Transformer [4], as a new deep learning technique, has been commonly used in natural language processing [5] as well as speech processing [6] and vision tasks [7]. Comparing with traditional deep learning techniques, Transformer has better abilities parsing deep semantic meaning of images [8]. In this work, we introduce Transformer into the field of Lane detection, which is self-attention to enhance the lane features, as shown in Figure 1. The proposed method we called CTLane, which not only inherits the strong ability

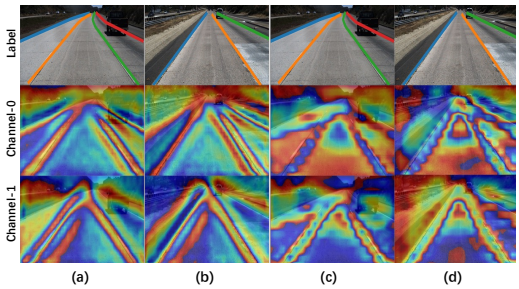

Fig. 1 The left side (a) and (b) shows the self-attention map on different channels, and the right side (c) and (d) shows the corresponding original feature map. The contrast shows that the lane features are significantly enhanced after our cnn-transformer module.

of extracting semantics from Transformer, but also leverages the high efficiency from conventional convolution. In several mainstream lane detection benchmark datasets, CTLane is evaluated, and show it can well maintain smoothness and continuity of lane detection with a high F1 score and a low false detection rate.

Our main contributions can be summarized as follows:

- We propose CNN-Transformer, which is used to extract high-level semantics of lanes. And we invent a new way to compute self-attention.
- We propose Fusion Decoder to aggregate high-level semantics and low-level local features, so as to better preserve the original features in images, and restore the information in decoders.
- We have achieved state of the art in accuracy on BDD100K dataset, and the tier 1 performance in Tusimple and CULane datasets.

2 Related work

For many years, lane detection has been a fundamental and important task in the field of autonomous driving, initially pioneered by feature-based lane detection methods [9, 10, 11], which first extract road features from images, and later apply Hough transform and Kalman filtering. However, the complexity of lane detection is also increasing, e.g., complex backgrounds and dynamic environment. With the surge of deep neural networks, the technology has push lane detection into a new height. [1]. In the current research on lane detection, since it lacks the uniform paradigm to describe the lane shape, the mainstream methods turned to take lane detection as a task of image segmentation [1, 2, 3], which is to distinguish the lanes from the background by semantic segmentation or instance segmentation.

Although, segmentation based methods have made great progress in lane detection so far, there are still many challenges such as illumination and occlusion [1]. To distinguish different lanes, most predict the probability of possible lanes pixel by pixel, and later connect the points belonging to the same lane by clustering, etc. [3, 12], but the aggregation process often increase the complexity of computation.

Since Transformer uses fully connected layers and weight matrix to propagate and transfer global information, it is difficult to directly embed the Transformers into the existing CNN architecture. In ResT[13], it is attempt to combine Transformer and CNN into an unified model, which gives some hints to our work.

3 Proposed method

Our overall model structure is shown in Figure 2.

3.1 Encoder

The encoder consists a backbone network, which is used to extract the lane features from the images. We use the ResNet-34 [14] as the backbone, and the output of the last convolution layer becomes the input of Feature Pyramid Network (FPN) [15], which is used to fuse the multi-scale features from the backbone, and the output of the FPN network becomes the input of the Convolution Attention (Convolution Transformer) Module we designed. While encoding, we keep the features of shallow layers in the feature extraction network to ensure that the decoder can receive the top level feature information.

3.2 CNN-Transformer

The detail structure of CNN-Transformer is shown in Figure 3.

3.2.1 Feature Interaction

Feature Interaction means features are fully blended into more a dense representation. In attention module, we need each pixel to contain global information, so we need to blend all feature

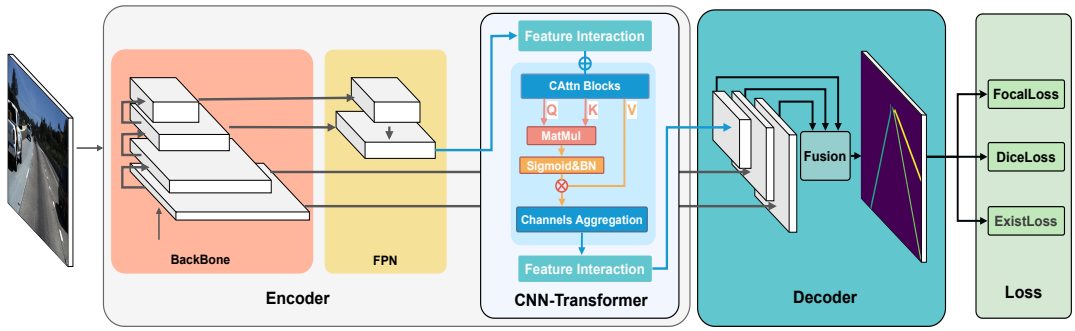

Fig. 2 The overall structure of the network is shown in the figure, and the network structure is mainly divided into three parts: encoder, CNN-Transformer and decoder. The encoder is used to extract the network to extract the image features, the CNN-Transformer is used to extract the deep semantic information, finally the decoder restore the feature map to the original input size.

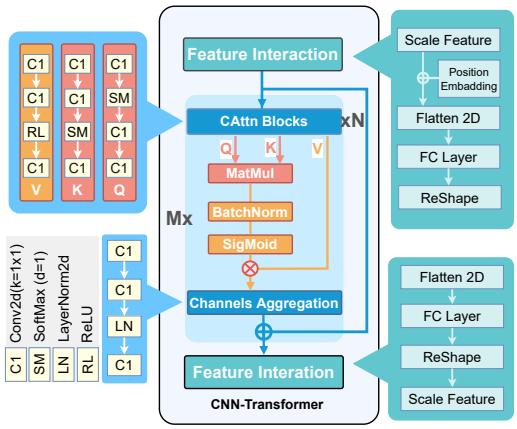

Fig. 3 The specific structure of the CNN-Transformer, in which the matrix used for mapping is replaced by different convolution blocks; the MHA does not scale channel of each Q, K, V, only aggregate all the channels after concat to the original channels through concat in the final stage; We also using sigmoid instead of softmax.

for each channels. In this sub-module, it is first to scale the input so that it maintains a moderate computational scale, then using Fully Connection (FC) layer and ReShape to reduce the feature into a new dimensional 2D matrix.

In Feature Interaction we can map the feature $M_{c\times h\times w}$ to $M_{c\times h'\times w'}$ in one FC layer. Therefore, to preserve the information of location, we create a positional code $P_{c\times h\times w}$ and embed the code into the original feature $M = M + P$. The position encode is introduced by using sine or cosine functions to add to the original signal.

Most Transformers applied in vision tasks normally cuts the input image up to a series of tokens, and is linearly mapped to the dimension $M_{n\times d}$, which breaks the original global spatial information. However, our directly uses the features extracted from the image, and preserve the spatial features from image as much as possible. Moreover, d is often larger than n for applying attention in vision, and our can effectively reduce the computational complexity of attention.

3.2.2 Convolution Attention Blocks

The self-attention operation firstly is to map the input matrix d_i to d_q, d_k, d_v respectively, where $d_q = d_k = d_v = d_m$. The input is usually linearly mapped into $M_{n\times d}$ and $M_{d\times d}$, and the complexity was $O(nd^2)$. However, in CTLane, we innovative replace the linear mapping operations in the attention block with Convolution Attention(CAttn) Blocks: $C^Q(\cdot), C^K(\cdot), C^V(\cdot)$. By leveraging 1×1 convolution, ReLU and Softmax, CAttn also can compute query, key and value matrices just equivalent to linear mapping version of Transformers.

In the CAttn blocks, we obtain the different Q, K, V by convolving the input. Turning input $I_{c\times h'\times w'}$ into output $Q_{c'\times h'\times w'}$, $K_{c\times h'\times w'}$, and $V_{c'\times h'\times w'}$ by convolution. After obtaining Q, K, V we can perform the attention calculation, though $Q\cdot K^T$ got $M_{c'\times h'\times h'}$, which we obtain the final self-attentive feature by $Sigmoid(\cdot)$ with $V_{c'\times h'\times w'}$ multiplying by each other to obtain the final self-attentive features: $O_{c'\times h'\times w'}$ The Attention function is

$$CAttn(Q,K,V) = Sigmoid(BN(\frac{QK^T}{\sqrt{d_m}}))V \quad (1)$$

where $BN(\cdot)$ is the Batch Normalization operation, and $Sigmoid(\cdot)$ is the Sigmoid function.

Since the Sigmoid requires the data distributed in appropriate range, a Batch Normalization is applied beforehand.

We here adopt CAttn, since the linear mapping usually is slow [8] when computing with large scale feature maps. Furthermore, 2D image features are needed to flatten into 1D features, which may lead to the destruction of spatial information [16]. Hence, we take advantage of the fact that different channels represent different image features, a linear mapping of each channel feature is performed using convolution. This gives us a more significant speedup than linearly mapping the entire information.

3.2.3 Multi-Head Attention

For multi-head attention can be defined as mapping Q with K to match V, with the dimensions d_q, d_k and d_v. Since it is multi-head self-attention $d_q = d_k = d_v = \frac{d_m}{t}$, t was number of head. We obtain the attention map by convolution with less computation, so we can use the complete channel to merge the multi-head attentions.

In the attention stage we use convolution to generate Q, K and V, so multi-head attention means that we only need to use different convolutions $C_i^Q(\cdot), C_i^K(\cdot), C_i^V(\cdot)$ to generate several different Q_i, K_i and V_i, $i \in \{0, 1, ..., t\}$. After obtaining Q_i, K_i and V_i, we can perform the attention calculation. When using multi-head attentions, we use $Concat(\cdot)$ to save the attention map obtained after each attention for the final aggregate attention map.

3.2.4 Channels Aggregation

Deal with attention feature maps generated by MHA, we need to use convolution to aggregate them together for subsequent computation. It is very simple and cheap to aggregate the attention maps obtained from different attention heads using convolution. We aggregate the multi-headed features $M_{c' \cdot t \times h' \times w \times h' \times w'}$ obtained from the previous step $Concat(\cdot)$ by using $Aggr(\cdot)$ and output as $M_{c \times h' \times w'}$. The $Aggr(\cdot)$ represents the Concatenation feature map with the dimension $c \cdot t \times h' \times w'$ is reduced to $c \times h' \times w'$ by 1×1 convolution and Layer Normalization. The all process can be described as:

$$
\begin{aligned}
&CMHA(x) = Aggr(Concat(L)),\\
&L = \{CAttn(C_i^Q(x), C_i^K(x), C_i^V(x)), i \in \{0, ..., t\}\}
\end{aligned}
\tag{2}
$$

3.3 Fusion decoder

The decoder component mainly reorganizes multiple feature maps by deformation, and reconstruct the segmentation output by reshaping. The detail structure of Fusion decoder is shown in Figure 4. Since deep networks may lose small or tiny targets during feature extraction, incorporating shallow features is to focus on on global information and prevent losing the type of features.

We use the tow features from lower layers in backbone $C_f \times H_1 \times W_1, C_f \times H_2 \times W_2$ and one features $C_3 \times H_3 \times W_3$ after CNN-Transformer. The number of all features channel would changed to C_f. Then the first and second lower layers with sizes respectively are scaled to the CNN-Transformer outputs feature size $H_3 \times W_3$ by linear interpolation. Finally the features F are fused by the concat function, with the shape $3C_f \times H_3 \times W_3$.

To make the decoder arbitrarily scaling, we add one Channels Scale layer. The feature map F is scaled to $S^2 \cdot N \times H_3 \times W_3$ then resize to $N \times H_o \times W_o$, where $C_i \times H_i \times W_i$, $0 < i < 4, i \in N$ denotes the number of feature channels in the ith layer. N denotes the number of output channels, H_o, W_o denotes the height and width of the output image, and S denotes the scaling multiplier to satisfy $S \cdot H_3 = H_o, S \cdot W_3 = W_o$.

3.4 Losses

In order to cope with real-life situations where there are intersections or partial turnoffs without lanes, we introduce an additional Exist Loss to determine whether lanes exist, determine whether the current lane exists a branch part of decoder, get the possibility of lanes in the current image then use Binary CrossEntropy Loss (BCE) to calculate the loss,where l_i is the target value of lane status, e_i is the softmax output.

$$
L_{Exist} = -\frac{1}{N} \sum_i [l_i \cdot e_i + (1 - l_i) \cdot (1 - e_i)] \tag{3}
$$

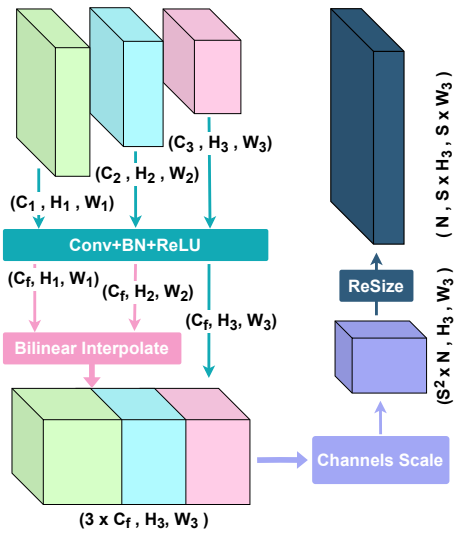

Fig. 4 The main structure of the decoder is shown in the figure, the shallow features obtained in the feature extraction layer and the deep features are sampled to the same size by dustample, and then the size is scaled to the appropriate size by convolution to scale the target size.

For our binary split branch, we use a weighted binary cross-entropy loss, and in order to better handle the split task of unbalanced examples, we use a loss function that is improved on the basis of Focal Loss [17]. The loss is calculated as

$$L_{Focal} = \frac{-1}{N}\sum_{i}[t_i^{\lambda} \cdot log(o_i) + (1-t_i)^{\lambda} \cdot log(1-o_i)] \tag{4}$$

where t_i is the target value of pixel i, o_i is the Softmax output, the focal coefficient $\lambda \geq 0$, and it becomes the standard cross-entropy loss function when $\lambda = 0$.

To solve imbalance between examples, the Dice loss [18] is used for the segmentation output:

$$L_{Dice} = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{2(1-o_i)t_i}{o_i + t_i}) \tag{5}$$

The weighted sum focal loss, dice loss and exist loss is then used as the total loss function of CTLane, where $\alpha, \beta, \gamma \geq 0$:

$$L = \alpha L_{Focal} + \beta L_{Dice} + \gamma L_{Exist} \tag{6}$$

4 Experiments and results

To compare the method we proposed, three widely used lane detection datasets are adopted in the experiment. They are CULane [1] and Tusimple Lane [19] and BDD100K[20].The CULane contains 55 hours of video, including urban and highway scenes, with 9 different scenes including Normal, Crowd, Curve, Dazzling Night, Night, etc. The Tusimple Lane was collected with stable light conditions on highways, where there are the major differences between the various types of data. BDD100K has a large number of different scenes, it contains road images with weather, scene, lighting and other factors.

4.1 Implementation

To augment training data, we use random Affine-transform, random horizontal flip, color shift and other techniques to generate more examples. In the Encoder, we use the pre-trained ResNet[14] and DLA[21] as the backbone to extract multi-scale features. In Tusimple, all images are zoomed into 352×640, then generated into in four scales: 88×160, 44×80, 22×40, and 11×20. The latter two feature maps are fused by FPN into a 22×40 feature map which then is transfer into CNN-Transformer using 3 head and attention 6 times. The fused feature map is fused again with the first two feature maps, and transfer into the Decoder to get the segmentation output with the size of 352×640.

The optimization function we used is SGD with learning rate set to 0.1, momentum set to 0.9, and weight decay set to 0.0001. The scheduler use Cosine Annealing with the step set to 5 and warmup set to 3. Tusimple lane has 200 epochs, culane 40 epochs, and BDD100K 60 epochs.In the combined loss function,$\gamma = 0.1$, $\alpha = 1.0$ and $\beta = 0.5$. We train the model on a NVIDIA 1080TI.

4.2 Evaluation metrics

In Tusimple, there are three official assessment measurement: accuracy, false positive (FP) and false negative (FN). The accuracy is defined as $accuracy = \frac{\sum_{clip} C_{clip}}{sum_{clip} S_{clip}}$. where C_{clip} is the number of correctly predicted lane points (predicated points within the range of 20 pixels around ground truth points), and S_{clip} is the total number of

Method	BackBone	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	CrossR	Night
Seg	ResNet101	70.80	90.20	68.20	59.80	64.60	41.70	84.10	65.50	1569	66.10
SCNN	VGG16	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10
UFLD	ResNet18	68.40	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10
UFLD	ResNet34	72.30	90.70	70.20	59.50	69.30	44.40	85.70	69.50	2370	66.70
SAD	ResNet34	70.70	89.90	68.50	59.90	67.70	42.20	83.80	66.00	1960	64.60
SAD	ResNet101	71.80	90.70	70.00	59.90	67.00	43.50	84.40	65.70	2052	66.30
PINet	Hourglass4	74.40	90.30	72.30	66.30	68.40	49.80	83.70	65.60	1427	67.70
RESA	ResNet34	74.50	91.90	72.40	66.50	72.00	46.30	88.10	68.60	1895	69.80
RESA	ResNet50	75.30	92.10	73.10	**69.20**	72.80	**47.70**	**88.30**	**70.30**	1503	69.90
Ours											
CTLane	ResNet34	74.36	91.43	72.69	67.05	67.84	44.43	87.54	67.39	**1266**	69.81
CTLane	DLA34	**75.39**	**92.24**	**73.48**	66.87	**74.18**	46.46	88.20	69.23	1672	**71.20**

Table 1 Comparison with state-of-the-art results on CULane dataset with IoU threshold = 0.5. For crossroad, only FP are shown.

Method	BackBone	**F1(%)**	**Acc(%)**	**FP(%)**	**FN(%)**
PolyLaneNet	EfficientNetB0	90.62	93.36	9.42	9.33
FastDraw	ResNet50	94.44	94.90	5.90	5.20
SCNN	VGG16	95.97	96.53	6.17	**1.80**
RESA	ResNet34	96.93	96.82	3.63	2.48
E2E	ERFNet	96.25	96.02	3.21	4.28
LaneATT	ResNet34	96.06	96.10	5.64	2.17
FOLOLane	ERFNet	96.59	**96.92**	4.47	2.28
CondLaneNet	ResNet34	96.98	95.37	2.20	3.84
PINet	Hourglass4	97.20	96.72	3.10	2.50
Ours					
CTLane	ResNet34	**97.54**	96.49	**2.01**	2.90
CTLane	DLA34	97.45	96.50	2.14	2.96

Table 2 Comparison results on Tusimple dataset.

ground truth points in each clip. However, Tusimple seems to become more saturated for many modern methods nowadays. Hence we add F1-score to evaluate the performance of the model, which is defined as $F1_{score} = \frac{2*precision*recall}{precision+recall}$, in which $precision = \frac{C_{clip}}{C_{clip}+F_{clip}}$ and $recall = \frac{C_{clip}}{C_{clip}+M_{clip}}$. F_{clip} is the number of lane points predicted incorrectly and M_{clip} is the number of ground truth points missed in each clip.

For the CULane dataset the official suggest to evaluate Precision, F1 and Recall. Each channel is treated as a 30-pixel-wide lane, using intersecting unions (IoU) to calculate predictions and ground truths. Where the predicted IoU is greater than the 0.5 threshold, it is marked as True Positives (TP). The evaluation function is defined as:

$F_1 = \frac{2\times Accuracy\times Recall}{Accuracy+Recall}$, $accuracy = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$.

4.3 Comparison

TUSimple We have conducted experiments several times on the Tusimple to show its performance. Since the accuracy in Tusimple is becoming saturated, we mainly focus on the F1 score. Table 2 shows CTLane with ResNet34 backbone

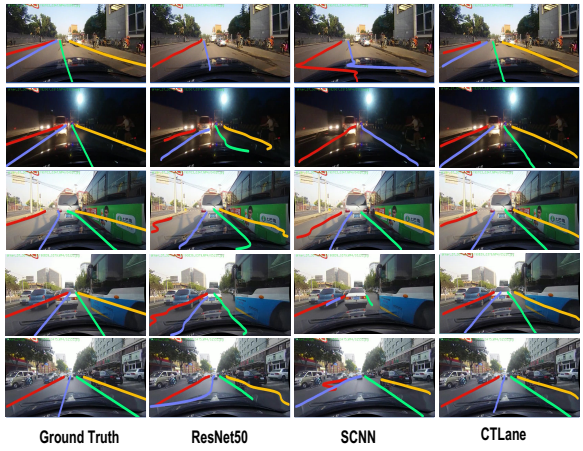

Fig. 5 Examples results from CULane dataset with ResNet50, SCNN and CTLane. It indicates that CTLane is immune to interference caused by dark night, shadows, strong light, etc.

made a significant improvement in F1 score compared to other methods, where the FP value outperforms better than others.

CULane For the more harsh dataset - CULane, the qualitative results for CULane are shown in Table 1. It indicate that some difficult scene with more severe occlusions, such as Crowded, Shadow, CrossRoad, Night, our method can still successfully infer the correct lanes. It is noticed that CTLane with DLA34 has achieved a large improvement in major scenarios as shown in Fig 5.

BDD100K The official lane segmentation results are given for both sides of the lane. We draw the complete lane mask for training and we keep the original lane in the val set. We use pixel

classification accuracy and lane IoU as evaluation metrics. Finally validated on the val set and output results on Table 3.

To explain the effectiveness of our method more visually, we show the qualitative results of our designed model and other models for the CULane dataset in Figure 5. Traditional lane detection methods cannot identify lane markings well in dark night, shadowed and strong lighted situations, resulting in the final prediction of the model breaking the continuity of the lanes. In contrast, our model can solve this problem well by attention. The results of our model show more robustness, and introducing our convolution attention rather than the traditional segmentation module can give the network a stronger ability to capture structured prior objects.

Our method shows in some occasions where the lanes are crowded and close to each other, it still separate the lanes well. As shown in Figure 5, even if the lanes in images are very close, our method still discriminate them successfully.

Method	BackBone	Acc(%)	IoU(%)
SCNN	VGG16	35.79	15.84
SAD	ResNet101	35.56	15.96
SAD	ENet	36.56	16.02
YOLOP	CSPDarknet	70.50	26.20
HybridNets	EfficientNetB3	85.40	**31.60**
Ours			
CTLane	ResNet34	84.64	26.12
CTLane	DLA34	**85.55**	26.68

Table 3 Comparison results on BDD100K dataset.

4.4 Ablation

Overall ablation study. We first investigated the effectiveness of the CNN-transformer and the fusion decoder component. As a baseline, we choose ResNet-34 as the backbone to extract features, and then we use a feature pyramid to aggregate multi-scale features to construct an encoder. We adopt deconvolution and bilinear interpolation in the decoder to up-sample the feature map and finally output the segmentation. We integrate the CNN-Transformer and Fusion Decoder respectively. The F1-score is summarized in the Table 4. We can see that both components greatly improve the lane detection performance, which proves the effectiveness.

BaseLine	CNN-Transformer	FusionDecoder	F1
✓			94.57
	✓		96.93
		✓	96.65
	✓	✓	**97.54**

Table 4 Experiments of the proposed modules on TuSimple dataset with ResNet-34 backbone.

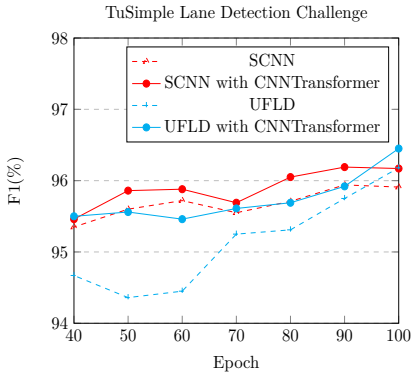

Fig. 6 CNN-Transformer applied On Different Models

Ablation study on generality of CNN-Transformer To validate the generalizability and stability of CNN-Transformer, we have implanted it into SCNN and UFLD. The Figure 6 shows that the model with the CNN-Transformer has a significant improvement on accuracy. It also indicates that CNN-Transformer helps convergence faster and is easily embedded into existing framework.

5 Conclusion

In this paper, we present a novel Lane detection method - CTLane with two portable network components: CNN-Transformers and Fusion Decoder. CNN-Transformer provides a novel convolution-based self-attention, which increases the speed by leveraging 1×1 convolution rather than matrix multiplication. CNN-Transformer also provides a solution to alleviate the problem of slow convergence of transformer and CNN mixing training. By applying the Fusion Decoder, the local features from the shallow layers are combined with the high-level semantics generated from CNN-Transformer. The extracted features are divided into a fusion layer and a deep feature layer, so that the network can process not only small local changes, but also take the lanes as a whole in images. The two components can be easily integrated into the existing framework arbitrarily. The

evaluation on three standard lane detection benchmark datasets indicates that CTLane is highly effective, generalized well and more stable.

Declarations

Ethical Approval

Not applicable

Competing interests

We declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Authors' contributions

Guoqiang Zhu and Mian Zhou wrote the main manuscript text and Zhouming Qi prepared all figures and Tables. All authors reviewed the manuscript.

Funding

This research is funded by National Natural Science Foundation of China, Grant No. 61971309.

Availability of data and materials

The authors confirm that the data and materials supporting the findings of this study are available within this paper and the repository https://github.com/squirtlecc/CNNTransformer.

References

[1] Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: Spatial CNN for traffic scene understanding 1712.06080

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

[3] Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Towards end-to-end lane detection: an instance segmentation approach 1802.05591

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need 1706.03762

[5] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North, pp. 4171–4186. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

[6] Dong, L., Xu, S., Xu, B.: Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5884–5888. IEEE. https://doi.org/10.1109/ICASSP.2018.8462506

[7] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) Computer Vision – ECCV 2020 vol. 12346, pp. 213–229. Springer. https://doi.org/10.1007/978-3-030-58452-8_13. Series Title: Lecture Notes in Computer Science

[8] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale 2010.11929

[9] Aly, M.: Real time detection of lane markers in urban streets. In: 2008 IEEE Intelligent Vehicles Symposium, pp. 7–12. IEEE. https://doi.org/10.1109/IVS.2008.4621152

[10] Gackstatter, C., Heinemann, P., Thomas, S., Klinker, G.: Stable road lane model based on clothoids. In: Meyer, G., Valldorf, J. (eds.) Advanced Microsystems for Automotive Applications 2010, pp. 133–143. Springer. https://doi.org/10.1007/978-3-642-16362-3_14

[11] Hur, J., Kang, S.-N., Seo, S.-W.: Multi-lane detection in urban driving environments using conditional random fields. In: 2013 IEEE Intelligent Vehicles Symposium (IV), pp. 1297–1302. IEEE. https://doi.org/10.1109/IVS.2013.6629645

[12] Ko, Y., Lee, Y., Azam, S., Munir, F., Jeon, M., Pedrycz, W.: Key points estimation and point instance segmentation approach for lane detection 2002.06604

[13] Zhang, Q., Yang, Y.: ResT: An efficient transformer for visual recognition 2105.13677

[14] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition 1512.03385

[15] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection 1612.03144

[16] Li, G., Xu, D., Cheng, X., Si, L., Zheng, C.: SimViT: Exploring a simple vision transformer with sliding windows 2112.13085

[17] Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection **42**(2), 318–327. https://doi.org/10.1109/TPAMI.2018.2858826

[18] Li, X., Sun, X., Meng, Y., Liang, J., Wu, F., Li, J.: Dice loss for data-imbalanced NLP tasks 1911.02855

[19] Tusimple: Tusimple lane detection benchmark

[20] Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., Darrell, T.: BDD100K: A diverse driving video database with scalable annotation tooling. CoRR **abs/1805.04687** (2018) 1805.04687

[21] Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep Layer Aggregation. arXiv. http://arxiv.org/abs/1707.06484 Accessed 2022-09-05