ITUJournal
Future and evolving technologies

ITU Journal on Future and Evolving Technologies, Volume 6, Issue 2, June 2025

ARTICLE

CTLane: An end-to-end lane detector by a CNN transformer and fusion decoder for edge computing

Mian Zhou[1], Guoqiang Zhu[3], Zhikun Feng[2], Haoyi Lian[1], Siqi Huang[1]

[1] School of AI and Advanced Computing, XJTLU Entrepreneur College (Taicang), Xi'an Jiaotong-Liverpool University, Suzhou 215412, China, [2] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, 610000, Chengdu, Sichuan, China, [3] Tianjin University of Technology, 300382, Tianjin, China

Corresponding author: Mian Zhou, Mian.Zhou@xjtlu.edu.cn

In advanced driving assistance systems and autonomous vehicles, lane detection plays a crucial role in ensuring the safety and stability of the vehicle during driving. While deep learning-based lane detection methods can provide accurate pixel-level predictions, they can struggle to interpret lanes as a whole in the presence of interference. To address this issue, we have developed a method that includes two components: a convolutional neural network transformer and a fusion decoder. The CNN transformer extracts the overall semantics of the lanes and speeds up convergence, while the fusion decoder combines high-level semantics with low-level local features to improve accuracy and robustness. By using these two components together, our method is able to effectively detect lanes in a variety of conditions, even when interference is present. We tested our method on multiple lane datasets and obtained superior results, with the best performance on the BDD100K dataset. Our method has successfully addressed the challenge of accurately and completely detecting lanes in the presence of interference, such as darkness, shadows, and strong light. The algorithm has been employed in an edge computing device, an intelligent cart. The code has been made available at: https://github.com/squirtlecc/CNNTransformer

Keywords: Convolution, deep learning, lane detection, transformer

1. INTRODUCTION

Lane detection is a crucial component of the perception phase in Advanced Driver Assistance Systems (ADAS) and autonomous driving systems, playing an essential role in ensuring vehicle safety and guiding driving paths. Unlike humans who rely on vision and experience to judge lanes, lane detection uses algorithms to accurately identify lane markings and road boundaries [1]. Vehicles acquire information about the road and its surroundings through visual sensors (e.g., cameras) and they feed this data into deep Convolutional Neural Networks (CNNs) to extract and analyse key features, providing reliable foundational data for subsequent path planning and driving decisions.

With the rapid advancement of deep learning, many research methods [2, 3, 4] treat lane detection as a segmentation task and employ end-to-end neural network frameworks. During segmentation, the model must focus on every pixel and predict its category. However, this pixel-wise processing makes it challenging for the model to treat the lane line as a whole, leading to a loss of lane semantics in feature maps. This problem is exacerbated under unfavourable lighting conditions, such as when shadows are cast on lanes, vehicles block the lanes, or during nighttime driving, significantly impacting model performance. Analysing lane feature maps generated by deep learning models reveals that the network does not always focus on the lane regions during feature extraction, which limits its detection performance. To address this, we propose introducing an attention mechanism to enhance the network's focus on key regions, thereby improving the overall performance of lane detection.

A transformer [5] is a deep learning technique that has been widely applied in natural language processing [6], speech processing [7], and vision tasks [8]. It excels at parsing deep semantic information from images [9], making it highly promising for lane detection tasks.

In recent years, a Vision Transformer (ViT) [9, 10] has achieved remarkable results in image classification tasks. Compared to traditional CNNs, ViT has a better ability to parse deep semantic information from images [9]. However, since a transformer uses fully connected layers and weight matrices to propagate and transfer global information, directly embedding a transformer into existing CNN architectures is difficult. ResT [11] was the first to combine a transformer and CNN into a unified model, and has provided some inspiration for our work.

Traditional transformers utilize large matrices as training parameters, leading to a significant number of parameters and slow network training convergence. Furthermore, the advantages of convolution are not fully exploited in hybrid models. To address these issues, we propose replacing matrices with convolutions for computing attention in feature maps, thereby reducing the number of parameters and overcoming the challenges of heavy training weights and slow training speeds associated with traditional transformers.

In this study, we propose a method called **CTLane**, which combines the powerful semantic extraction capability of transformers with the efficiency of traditional CNNs. By introducing CNNs into the structure of transformers, the model not only significantly improves training speed but also enhances generalization ability. The CTLane model incorporates a multi-head attention mechanism to effectively extract features in the image space, enabling the model to focus more precisely on global lane features in the feature map, thereby predicting more complete, smoother, and continuous lane lines, as shown in Fig. 1. We conducted a comprehensive evaluation of CTLane on several benchmark lane detection datasets, and the results demonstrate that the model maintains high lane detection accuracy while achieving a higher F1 score and a lower false detection rate.

Our main contributions are summarized as follows:

- We propose a CNN transformer to extract high-level semantics of lanes and introduce a novel method to compute self-attention.
- A CNN transformer combines the advantages of a CNN and transformer to better aggregate spatial features and can be easily deployed after the feature extraction stage of any CNN.
- We propose the fusion decoder, which aggregates high-level semantic features and low-level local features, effectively preserving the original features in images and restoring information in the decoder.
- We achieve state-of-the-art accuracy on the BDD100K dataset and tier-1 performance on the Tusimple and CULane datasets.

2. RELATED WORK

Within autonomous driving and ADAS applications, lane departure is one of the main causes of traffic accidents, highlighting the importance of lane detection. As the rapid development of deep learning technology, this detection has gradually shifted from traditional feature-based methods to deep learning-based methods [12]. Traditional lane detection methodologies predominantly depend on manually designed feature extraction techniques, such as color segmentation, texture analysis, and edge detection, and they subsequently employ post-processing techniques like the Hough transform or Kalman filtering to extract lane lines. However, these methods perform poorly in complex scenarios, such as changes in lighting, occlusion, and complex road structures.
Upon reviewing a substantial body of literature, we have observed that the transformer [13] is a novel deep learning technique commonly employed in natural language processing [14] and speech processing [15]. Unlike traditional CNN, the transformer, through its self-attention mechanism, is capable of capturing global information within images, thereby excelling in contextual modeling and the capture of long-range dependencies. Several studies [16] [17] have significantly enhanced the global modeling capabilities of lane detection by incorporating a transformer architecture. For example, by generating conditional convolutional kernel parameters and integrating a row-by-row classification strategy, these studies have achieved high-precision and efficient lane detection. In ResT [18], the first attempt to combine a transformer and CNN into a unified model has provided valuable insights for our research.
Currently, the main methods in lane detection are deep learning-based methods. They can be divided into four categories: semantic segmentation, row-wise classification, anchor, and curve fitting.

2.1 Semantic segmentation-based methods

Lane detection methods based on semantic segmentation achieve precise differentiation between lanes and background by transforming the task into a pixel-level classification problem and leveraging deep neural networks. Typical approaches, such as UNet and its variants[19] [20], employ encoder-decoder architectures combined with multiscale feature fusion to enhance accuracy. ENet-21[21] introduces lightweight convolutions and affinity field techniques, maintaining high performance while reducing model complexity. GANs further improve feature extraction capabilities through adversarial learn-

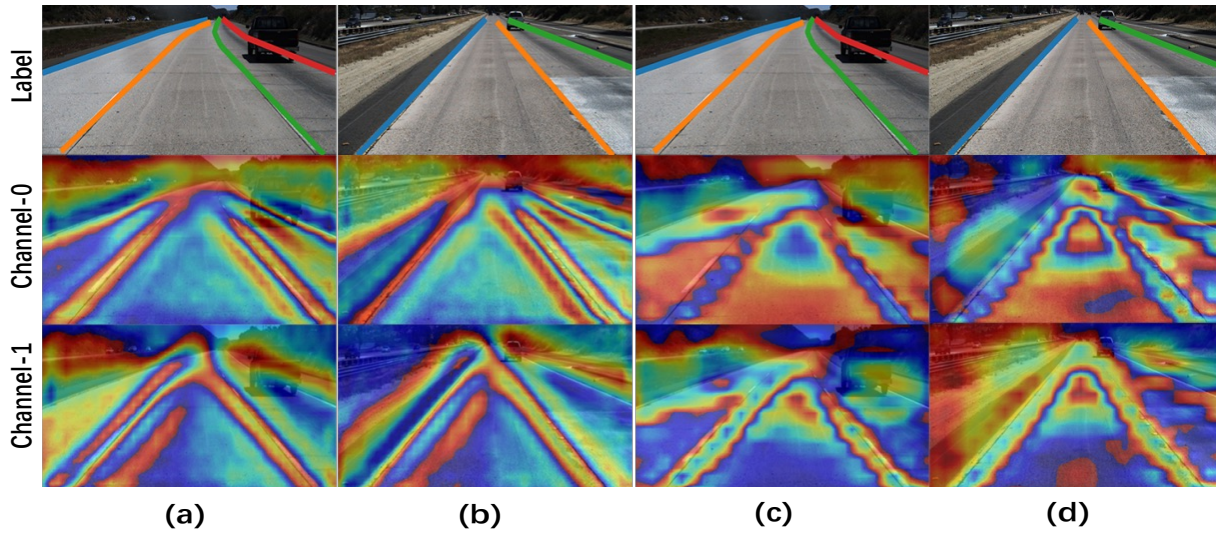

Figure 1 – The left-side (a) and (b) show the self-attention maps of the CTLane method on different channels, while the right-side (c) and (d) display the corresponding original feature maps. The comparison demonstrates that, under the influence of our CNN-Transformer module, the lane features are significantly enhanced, leading to the prediction of more complete, smooth, and continuous lane lines.

ing [22]. To address domain adaptation challenges, the MLDA framework [23] optimizes at pixel, instance, and category levels, while the integration of spatiotemporal information [24] enhances detection stability through hybrid spatiotemporal architectures. Additionally, the combination of semantic segmentation and anchor-based detection [25] achieves superior generalization and real-time performance in multi-task perception.

2.2 Row-wise classification-based methods

Lane detection processes have been further streamlined by the classic CNN row-by-row classification method [26], which handles image features on a per-row basis. High-precision and efficient lane detection has been achieved by some studies [16] [27] through the generation of conditional convolutional kernel parameters and the integration of a row-by-row classification strategy. GroupLane [28] has implemented efficient 3D lane detection by employing a channel grouping strategy and a row classification head design, combined with Birds-Eye View (BEV) features and a Self-Organizing Map (SOM) mechanism.

2.3 Anchor-based methods

Anchor-based methods extract lane features through predefined anchors and generate candidate lanes based on these anchors, significantly improving efficiency and accuracy. For instance, some studies utilize anchor-chain representations to model lanes and enhance the model's perception of lane instances through multi-reference deformable attention [29]. Furthermore, in order to reduce computational costs, the introduction of local and global polar coordinate modules decreases the number of anchors, while the triplet detection head enables end-to-end detection without NMS, enhancing performance in dense scenarios [17]. In 3D lane detection, the definition of 3D anchors combined with iterative regression and global optimization avoids the complexity of traditional bird's-eye view transformations [30]. Hybrid anchor-driven ordinal classification further reduces computational costs and improves localization accuracy [31].

2.4 Curve fitting-based methods

Lane detection methods are based on curve fitting model lanes as continuous curves. For instance, PolyLaneNet [32] utilizes deep polynomial regression to directly output the polynomial coefficients of lane markings, achieving accuracy comparable to existing methods while maintaining a high frame rate (115 FPS). Another study [33] enhances global context modeling through a transformer network, directly regressing a parameterized lane shape model and thus avoiding the intermediate segmentation steps and post-processing involved in traditional methods. Furthermore, the selective focus framework [34] introduces a lane distortion score to quantify the impact of quantization errors, thereby further enhancing detection performance.

3. PROPOSED METHOD

The model is described as follows: Given an input image $I \in R^{C\times H\times W}$, the goal of CTLane is to output lanes $I \in R^{N\times H\times W}$, where N denotes the maximum number of scheduled lanes. Our overall model structure is shown in Fig. 2. The encoder consists of a backbone network that is used to extract the lane features from the images. We use ResNet-34 [35] as the backbone, and the output of the last convolution layer becomes the input of a Feature Pyramid Network(FPN) [36], which is used to fuse the multiscale features from the backbone, and the output of the FPN network becomes the input of the convolution attention (convolution transformer) module we designed. The convolution attention (convolution transformer) module is to further process high-level features from FPN. In the decoder, the shallow semantics and the deeper information are fused together. The final feature map of lane segmentation is to predict the presence of each channel and probability distribution, then perform binary classification. This section shows the details of our design model; firstly, there is an overview of the model structure followed by a detailed presentation of the attention module of the volume machine and its decoder part, respectively.

3.1 Encoder

An encoder consists of a backbone network that is used to extract the lane features from the images. We use ResNet-34 [35] as the backbone, and the output of the last convolution layer becomes the input of the Feature Pyramid Network (FPN) [36], which is used to fuse the multiscale features from the backbone, and the output of the FPN network becomes the input of the convolution attention (convolution transformer) module we designed. During 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

Due to the limitation of convolution operations to local receptive fields, traditional feature representations make each pixel's features rely solely on its local region, lacking global contextual information. This limitation hinders the effective modeling of relationships between different parts of the image.

To address these issues, we propose a feature interaction mechanism. This mechanism fully integrates features within each channel, transforming them into more compact and globally enriched representations, enabling each pixel to incorporate global contextual information. Feature interaction means features are fully blended into a more dense representation. In the attention module, we need each pixel to contain global information, so we need to blend all the features for each channel. In this submodule, the main task is to first scale the input so that it maintains a moderate computational scale, then to use the Fully Connection (FC) layer and ReShape to reduce the feature into a new dimensional 2D matrix.

In the 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 location information, we create a positional code $P_{c\times h\times w}$ and embed the code into the original feature $M = M + P$. The position encoder is introduced by using sine or cosine functions to add to the original signal.

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

In the original transformer [5], to keep position information between word embedding in the mapping process, position encoding is introduced by using sine or cosine functions to construct the value of each position so that it encodes the positional relationship into embedding in the computational process. As its extension in vision, ViT [9] uses patches as the latitude of image tokens, encoding the position of each token in such a way that the corresponding position information is maintained when projected onto a 2D image.

As shown in Fig. 3, at the end of the CNN transformer structure, we use similar operations to map the features back to the original dimensions.

3.2.1 Convolution attention blocks

The traditional transformer method typically divides the input image into a series of tokens and flattens them into one-dimensional vectors to compute the global relationships among all features.However, this approach ignores the two-dimensional spatial structure of the image, which can lead to the loss of spatial information [37]. In the self-attention operation, the input matrix $\mathbf{M}_{n\times d_{\text{in}}}$ is first linearly transformed to generate the query matrix $\mathbf{Q}_{n\times d_m}$, key matrix $\mathbf{K}_{n\times d_m}$, and value matrix $\mathbf{V}_{n\times d_m}$, where $d_q = d_k = d_v = d_m$. However,in CTLane, we replace the linear mapping operations in the attention block with Convolution Attention (CAttn) blocks using $C^Q(\cdot), C^K(\cdot), C^V(\cdot)$, which compute query, key, and value matrices equivalent to the linear mapping version

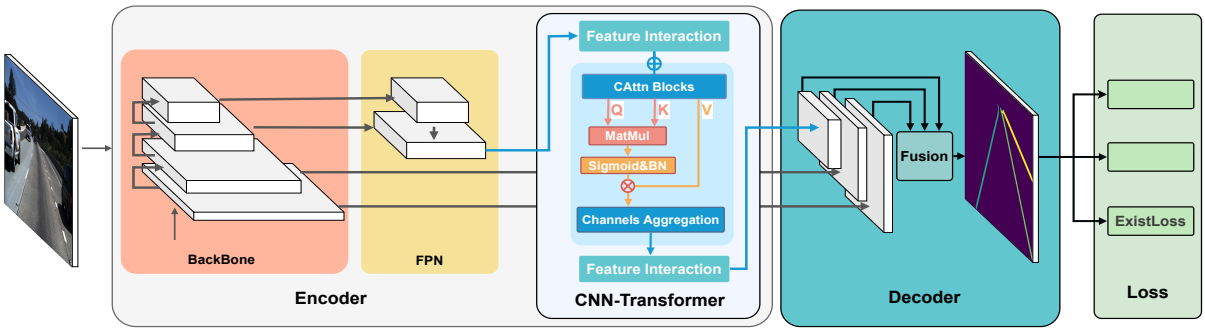

Figure 2 – The overall structure of the network is shown in the figure, and the network structure is mainly divided into three parts: encoder, CNN transformer and decoder. The encoder is used to extract the network to extract the image features, the CNN transformer is used to extract the deep semantic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

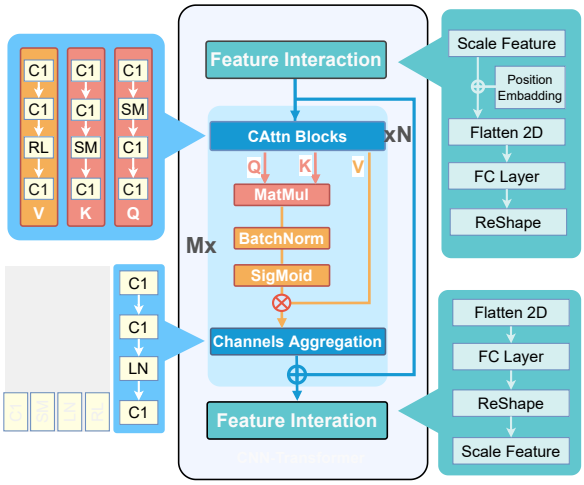

Figure 3 – The CNN transformer has a specific structure in which the mapping matrix is replaced by different convolution blocks. The MHA does not scale the channels of each Q, K, V, but rather concatenates all the channels to the original channels through concatenation in the final stage. We also use sigmoid instead of softmax.

of transformers through 1×1 convolution, ReLU, and softmax. In the CAttn block, the input $M_{c\times h'\times w'}$ is transformed into the outputs $Q_{c'\times h'\times w'}$, $K_{c\times h'\times w'}$, and $V_{c'\times h'\times w'}$ through convolution. We compute the dot product of the query with all keys, divide each dot product result by $\sqrt{h' \cdot w'}$, introduce batch normalization to normalize the computed attention scores, and then apply the sigmoid function to obtain the weights for the values. We calculate the output matrix with the following formula:

$$CAttn(Q,K,V) = Sigmoid(BN(\frac{QK^T}{\sqrt{h' \cdot w'}}))V, \qquad (1)$$

where $BN(\cdot)$ is the batch normalization operation, and $Sigmoid(\cdot)$ is the sigmoid function.

In the CAttn block, 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 speed-up than linearly mapping the entire information.

3.2.2 Multi-Head attention

In traditional multi-head self-attention mechanisms, the query, key, and Value (V) are typically mapped to different feature spaces through linear projections. Parallel attention computations are then performed to capture information from different subspaces. However, each head in MHA will only pay attention to its own subspace, which makes attention heads neglect each other. Our method obtains the attention map through convolution with reduced computational cost, enabling the use of the complete channel to effectively merge the multi-head attentions.

Our approach introduces convolution operations to generate Q, K, and V. In the multi-head attention mechanism, we utilize different convolution operations $C_i^Q(\cdot)$, $C_i^K(\cdot)$, and $C_i^V(\cdot)$ to generate multiple heads of Q_i, K_i, and V_i, where $i \in 1, 2, \ldots, t$, and t represents the number of attention heads. After obtaining Q_i, K_i, and V_i for each head, we perform parallel attention computations for each set and concatenate the results of all heads using the $Concat(\cdot)$ operation, ultimately producing the aggregated attention map. The computational process can be expressed by the following formula:

$$\begin{aligned} CMultiHead(\hat{M}) &= Concat(head_1, \ldots, head_t), \\ head_i &= CAttn(C_i^Q(\hat{M}), C_i^K(\hat{M}), C_i^V(\hat{M})), \end{aligned} \qquad (2)$$

where t is the number of attention heads.

3.2.3 Channels aggregation

To process the attention feature maps generated by MHA, we need to use convolution to aggregate them for subsequent computation. It is simple and efficient 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 using $Concat(\cdot)$ and output them as $M_{c\times h'\times w'}$. The function $Aggr(\cdot)$ represents the concatenation feature map, with the dimension $c \cdot t \times h' \times w'$ reduced to $c \times h' \times w'$ by 1×1

convolution and layer normalization. The entire process can be described as:

$$CMHA(x) = Aggr(CMultiHead(\hat{M})). \quad (3)$$

3.2.4 Time complexity analysis of MHA and CMHA

In the MHA mapping stage, $M_{n\times d}$ is mapped to $M_{n\times d}$ with a $k\times k$ convolution kernel, and its total complexity is evaluated by $O(n^2k^2d)$. In CT, we uses 1×1 kernel instead of $k\times k$ for the mapping operation, so that the mapping complexity is $O(n^2d)$. The MHA and Concat matrix $M_{nt\times d}$ is compressed into $M_{n\times d}$, and the complexity is $O(tn^2d)$. Hence, the total compressing the multi-headed attention Concat matrix as $M_{nt\times d}$ to $M_{n\times d}$ by total convolution, its complexity is $O(tn^2d)$, so the total complexity is $O(n^2d + n^2d)$. In this case, $d > n$, the $CMHA(\cdot)$ complexity is lower than $MHA(\cdot)$.

In comparison to the number of training parameters required, the number of training parameters required by the traditional transformer to generate Q, K, V for the input features $F = M_{1\times l}$ and once attention is $3\times l\times d$, while the number of parameters required to generate Q, K, V with 1×1 convolution is $3\times n\times1$, where n denotes the times of convolutions used in Q, K, V.

In addition, the $CMHA(\cdot)$ is more suitable for the input as images, because the image is a 2D matrix, and the convolution can be used to extract the spatial features from images, which is more suitable for the attention mechanism.

3.3 Fusion decoder

The decoder component mainly reorganizes multiple feature maps by deformation and it reconstructs the segmentation output by reshaping. The detailed structure of a fusion decoder is shown in Fig. 4. Since deep networks may lose small or tiny targets during feature extraction, incorporating shallow features focuses on global information and prevents losing these 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 the CNN transformer. The number of all features of a channel would change 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 scale, we add one channel 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 $S\cdot H_3 = H_o, S\cdot W_3 = W_o$.

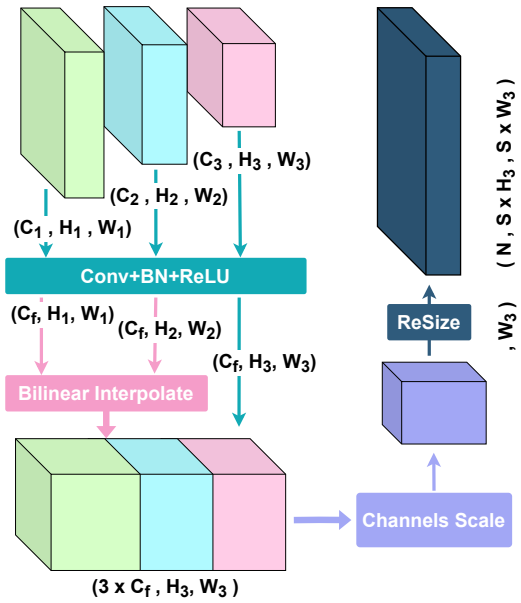

Figure 4 – The main structure of the decoder is shown in the figure; the shallow features obtained in the feature extraction layer and the deep features are sampled to the same size by dustample, and then the size is scaled to the appropriate size by convolution to scale the target size.

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 has a branch part of a 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.

$$L_{Exist} = -\frac{1}{N}\sum_i [l_i\cdot e_i + (1-l_i)\cdot(1-e_i)] \quad (4)$$

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 [38]. The loss is calculated as

$$L_{Focal} = -\frac{1}{N}\sum_i [t_i^{\lambda}\cdot log(o_i) + (1-t_i)^{\lambda}\cdot log(1-o_i)] \quad (5)$$

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 [39] is used for the segmentation output:

$$L_{Dice} = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{2(1-o_i)t_i}{o_i + t_i}) \quad (6)$$

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

$$L = \alpha L_{Focal} + \beta L_{Dice} + \gamma L_{Exist} \quad (7)$$

4. EXPERIMENTS AND RESULTS

To compare the method we proposed, three widely used lane detection datasets are adopted in the experiment. They are CULane [2], Tusimple Lane [40], and BDD100K [41]. The CULane contains 55 hours of video, including urban and highway scenes, with nine different scenes including normal, crowd, curve, dazzling night, night, etc. The Tusimple Lane was collected with stable light conditions on highways, where there are 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.

The details of the datasets are shown in Table 1.

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 [35] and DLA [48] as the backbone to extract multiscale 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 transferred into a CNN transformer using 3 head and attention 6 times. The fused feature map is fused again with the first two feature maps, and transferred into the decoder to get the segmentation output with the size of 352×640.

The optimization function we used is Stochastic Gradient Descent (SGD) with learning rate set to 0.1, momentum set to 0.9, and weight decay set to 0.0001. The scheduler uses 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 NVIDIA 1080TI and 4080.

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 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

$$F1_{score} = \frac{2 \times precision \times recall}{precision + recall},$$

in which precision and recall are defined

$$precision = \frac{C_{clip}}{C_{clip} + F_{clip}}$$

$$recall = \frac{C_{clip}}{C_{clip} + M_{clip}}$$

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 suggestion is to evaluate precision, F1, and recall. Each channel is treated as a 30-pixel-wide lane, using Intersecting Unions (IoUs) to calculate predictions and ground truths. Where the predicted IoU is greater than the 0.5 threshold, it is marked as a True Positive (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 3 shows CTLane with ResNet34 backbone 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 are shown in Table 2. It indicate that for

Table 1 – Dataset type

Dataset	Scenario	Road Type	Frames	Train	Resolution	Max Lane
Tusimple	light traffic, day	highway	6,408	3,236	1280×720	5
CULane	night, day, traffic	urban, rural and highway	133,235	88,880	1640×590	4
BDD100K	light traffic, day	highway	100,042	58,269	1280×717	4

Table 2 – Comparison with state-of-the-art results on CULane dataset with IoU threshold = 0.5. For crossroad, only FP are shown.

Type	BackBone	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	CrossR	Night
UFLD[31]	ResNet34	72.30	90.70	70.2 0	59.50	69.30	44.40	85.70	69.50	2037	66.70
SCNN[12]	VGG16	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10
MCA-UFLD[42]	ResNet18	69.36	88.90	67.28	55.79	63.87	39.75	82.50	56.26	1741	63.30
STLNet[43]	Swin	73.60	91.80	70.20	65.90	69.30	48.80	85.30	67.50	68.20	
SAD[44]	ResNet101	71.80	90.70	70.00	59.90	67.00	43.50	84.40	65.70	2052	66.30
PINet[45]	Hourglass4	74.40	90.30	72.30	66.30	68.40	**49.80**	83.70	65.60	1427	67.70
E2E[26]	ERFNet	74.00	91.00	73.10	64.50	74.10	46.60	85.80	**71.90**	2022	67.90
RESA[46]	ResNet50	75.31	92.10	73.10	**69.20**	72.80	47.70	**88.30**	70.30	1503	69.90
LaneATT[47]	ResNet18	75.13	91.17	72.71	65.82	68.03	49.13	87.82	63.75	1020	68.58
Ours											
CTLane	ResNet34	74.31	91.49	72.68	67.31	68.33	45.27	87.60	68.94	1542	69.69
CTLane	DLA34	**75.39**	**92.24**	**73.48**	66.87	**74.18**	46.46	88.20	69.23	1672	**71.20**

Table 3 – Comparison results on TuSimple dataset.

Method	BackBone	**F1(%)**	**Acc(%)**	**FP(%)**	**FN(%)**
PolyLaneNet[32]	EfficientNetB0	90.62	93.36	9.42	9.33
FastDraw[49]	ResNet50	94.44	94.90	5.90	5.20
SCNN[12]	VGG16	95.97	96.53	6.17	**1.80**
RESA[50]	ResNet18	96.84	96.84	3.25	2.67
E2E[26]	ERFNet	96.25	96.02	3.21	4.28
LaneATT[32]	ResNet34	96.06	96.10	5.64	2.17
FOLOLane[51]	ERFNet	96.59	**96.92**	4.47	2.28
CondLaneNet[52]	ResNet101	97.24	96.54	2.01	3.50
Ours					
CTLane	ResNet34	**97.54**	96.49	**2.01**	2.90
CTLane	DLA34	97.45	96.50	2.14	2.96

some difficult scenes 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 are shown in Table 4.

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 Fig. 5. Traditional lane detection methods cannot identify lane markings well in dark night, shadowed and strong light 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 that on some occasions where the lanes are crowded and close to each other, it still separates the lanes well. As shown in Fig. 5, even if the lanes in images are very close, our method still distinguishes them successfully.

Table 4 – Comparison results on BDD100K dataset.

Method	BackBone	Acc(%)	IoU(%)
SCNN[12]	VGG16	35.79	15.84
SAD[44]	ENet	36.56	16.02
HWLane[53]	ResNet34	73.93	33.25
YOLOPv8[54]	CSPDarknet	84.90	28.80
HybridNets[55]	EfficientNetB3	85.40	**31.60**
Ours			
CTLane	ResNet34	84.64	26.12
CTLane	DLA34	**85.55**	26.68

4.4 Ablation

We have designed a series of ablation experiments to analyse the effectiveness of different components in our model.

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 multiscale features to construct

Table 5 – Experiments of the proposed modules on TuSimple dataset with ResNet-34 backbone.

Baseline	**CNN transformer**	**Fusion decoder**	**F1**
✓			94.57
	✓		96.93
		✓	96.65
	✓	✓	**97.54**

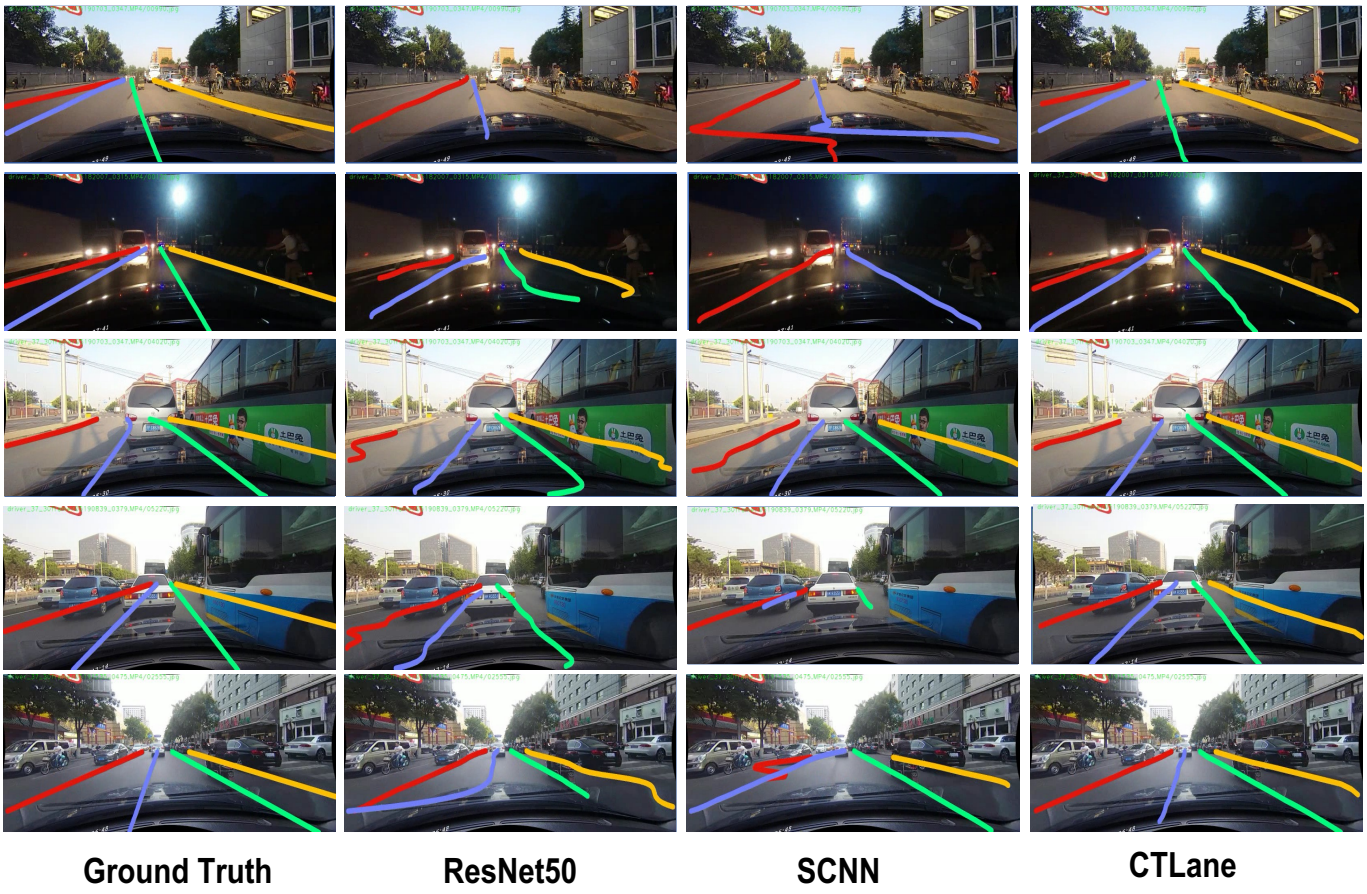

Figure 5 – Example results from CULane dataset with ResNet50, SCNN and CTLane. It indicates that CTLane is immune to interference caused by dark night, shadows, strong light, etc.

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 Table 5. We can see that both components greatly improve the lane detection performance, which proves the effectiveness.

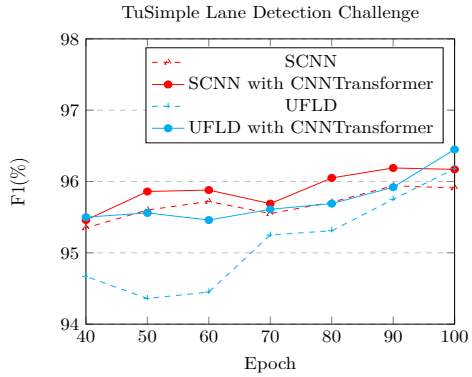

Figure 6 – CNN transformer applied on different models

Ablation study on generality of CNN transformer To validate the generalizability and stability of CNN transformer, we have implanted it into Spatial Convolutional Neural Network (SCNN) and Ultra Fast structure-aware deep Lane Detection (UFLD). The Fig. 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.

4.5 Edge computing deployment

To demonstrate the practicality and efficiency of CTLane in real-world automotive applications, we deployed the lane detection model on the iFlytek U-car as shown in Fig. 7, an intelligent vehicle equipped with an NVIDIA Jetson Nano edge-computing device. The Jetson Nano, with its compact design and energy-efficient performance, is well-suited for embedded systems in autonomous driving and Advanced Driver Assistance Systems (ADAS). This deployment aimed to evaluate the model's real-time performance, resource efficiency, and detection accuracy in a practical automotive environment. **Implementation Details**

- **Hardware setup**: The iFlytek U-car is powered by an NVIDIA Jetson Nano, featuring a 128-core Maxwell GPU and a quad-core ARM CPU. This hardware con-

Figure 7 – The iFlytek U-car equipped with edge-computing capabilities powered by the NVIDIA Jetson Nano is used by us in real-time perception on lane marks.

figuration provides a balance of computational power and energy efficiency, making it ideal for edge-based lane detection tasks.

- **Software environment**: The model was optimized using TensorRT, NVIDIA's high-performance deep learning inference library, to maximize inference speed and minimize memory usage. The framework was implemented in PyTorch, and the model was quantized to FP16 precision to further enhance computational efficiency without compromising detection accuracy.
- **Input resolution**: To ensure real-time performance on the Jetson Nano, the input resolution was set to 352×640, striking a balance between detection accuracy and computational load.

Performance metrics inference speed: On the Jetson Nano, the model achieved an average inference speed of 8 – 10 FPS (frames per second), close to meeting the real-time requirements for lane detection in automotive applications.

Resource utilization: The GPU utilization remained below 75%, indicating that the model is lightweight and leaves sufficient computational resources for other concurrent tasks, such as object detection or path planning.

Real-world testing We conducted extensive real-world testing on urban roads and highways using the iFlytek U-car to evaluate the model's performance under diverse driving conditions. The results showed that CTLane performs exceptionally well in challenging scenarios, including low-light environments, shadows, and occlusions caused by other vehicles or road obstacles. The model consistently maintained lane continuity and accuracy, proving its reliability for real-world deployment in ADAS and autonomous driving systems.

5. CONCLUSION

In this paper, we present CTLane, a novel lane detection method that integrates two key components: CNN transformer and fusion decoder. The CNN transformer introduces a convolution-based self-attention mechanism, which significantly improves computational efficiency by leveraging 1×1 convolutions instead of traditional matrix multiplications. This approach not only accelerates training convergence but also addresses the challenges of integrating transformer and CNN architectures. The fusion decoder effectively combines low-level local features from shallow layers with high-level semantic information generated by the CNN transformer. This dual-layer fusion enables the network to capture both fine-grained details and global lane structures, enhancing the robustness and accuracy of lane detection.

The proposed method demonstrates strong generalization capabilities and can be seamlessly integrated into existing frameworks. Extensive experiments on three benchmark datasets, CULane, Tusimple, and BDD100K show that CTLane achieves close to state-of-the-art performance, particularly in challenging scenarios such as low-light conditions, shadows, and occlusions. The results highlight the method's ability to maintain lane continuity and accuracy even under adverse conditions. To further validate the practicality of CTLane, we deployed the model on the iFlytek U-car, an intelligent vehicle platform equipped with an NVIDIA Jetson Nano edge-computing device. The deployment demonstrated the model's close to real-time performance.

Future work will focus on further optimizing the model for real-time applications and exploring its potential in other computer vision tasks. To enhance real-time performance, we will investigate dynamic network pruning, hybrid precision quantization, and hardware-customized acceleration strategies. Additionally, we aim to extend the framework to multi-task scenarios and cross-modal systems (e.g., fusing camera and LiDAR inputs). Improving robustness under extreme conditions (e.g., fog, heavy rain) through adversarial training or synthetic data augmentation will also be prioritized. The CTLane framework provides a robust foundation for advancing lane detection technologies, contributing to the development of safer and more reliable autonomous driving systems.

REFERENCES

[1] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. "Recent progress in road and lane detection: a survey". In: *Machine Vision and Applications* 25.3 (Apr. 2014), pp. 727–745. ISSN: 0932-8092, 1432-1769. DOI: 10.1007/s00138-011-0404-2. (Visited on 04/04/2022).

[2] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Spatial As Deep: Spatial CNN for Traffic Scene Understanding". In: *AAAI Conference on Artificial Intelligence*. 2017. URL: https://api.semanticscholar.org/CorpusID:9164115.

[3] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. "RESA: Recurrent Feature-Shift Aggregator for Lane Detection". In: *arXiv:2008.13719 [cs]* (Mar. 25, 2021). arXiv: 2008.13719. (Visited on 04/04/2022).

[4] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. "Towards End-to-End Lane Detection: an Instance Segmentation Approach". In: *arXiv:1802.05591 [cs]* (Feb. 15, 2018). arXiv: 1802.05591. (Visited on 04/04/2022).

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *arXiv:1706.03762 [cs]* (Dec. 5, 2017). arXiv: 1706.03762. (Visited on 04/04/2022).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North*. Proceedings of the 2019 Conference of the North. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. (Visited on 04/04/2022).

[7] Linhao Dong, Shuang Xu, and Bo Xu. "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Calgary, AB: IEEE, Apr. 2018, pp. 5884–5888. ISBN: 978-1-5386-4658-8. DOI: 10.1109/ICASSP.2018.8462506. (Visited on 04/04/2022).

[8] Andrew Brown, Cheng-Yang Fu, Omkar Parkhi, Tamara L Berg, and Andrea Vedaldi. "End-to-end visual editing with a generatively pre-trained artist". In: *European Conference on Computer Vision*. Springer. 2022, pp. 18–35.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv:2010.11929 [cs]* (June 3, 2021). arXiv: 2010.11929. (Visited on 04/04/2022).

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12346. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 213–229. ISBN: 978-3-030-58451-1 978-3-030-58452-8. DOI: 10.1007/978-3-030-58452-8_13. (Visited on 04/04/2022).

[11] Qinglong Zhang and Yubin Yang. "ResT: An Efficient Transformer for Visual Recognition". In: *arXiv:2105.13677 [cs]* (Oct. 14, 2021). arXiv: 2105.13677. (Visited on 04/04/2022).

[12] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Spatial as deep: Spatial cnn for traffic scene understanding". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[15] Linhao Dong, Shuang Xu, and Bo Xu. "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition". In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2018, pp. 5884–5888.

[16] Long Zhuang, Tiezhen Jiang, Meng Qiu, Anqi Wang, and Zhixiang Huang. "Transformer generates conditional convolution kernels for end-to-end lane detection". In: *IEEE Sensors Journal* (2024).

[17] Shengqi Wang, Junmin Liu, Xiangyong Cao, Zengjie Song, and Kai Sun. "Polar R-CNN: End-to-End Lane Detection with Fewer Anchors". In: *arXiv preprint arXiv:2411.01499* (2024).

[18] Qinglong Zhang and Yu-Bin Yang. "Rest: An efficient transformer for visual recognition". In: *Advances in neural information processing systems* 34 (2021), pp. 15475–15485.

[19] Der-Hau Lee and Jinn-Liang Liu. "End-to-end deep learning of lane detection and path prediction for real-time autonomous driving". In: *Signal, Image and Video Processing* 17.1 (2023), pp. 199–205.

[20] P Santhiya, Immanuel JohnRaja Jebadurai, Getzi Jeba Leelipushpam Paulraj, A Jenefa, S Kiruba Karan, et al. "Deep Vision: Lane Detection in ITS: A Deep Learning Segmentation Perspective". In: *2024 Second International Conference on Inventive Computing and Informatics (ICICI)*. IEEE. 2024, pp. 21–26.

[21] Seyed Rasoul Hosseini, Hamid Taheri, and Mohammad Teshnehlab. "Enet-21: an optimized light cnn structure for lane detection". In: *arXiv preprint arXiv:2403.19782* (2024).

[22] Swati Jaiswal and B Chandra Mohan. "Deep learning-based path tracking control using lane detection and traffic sign detection for autonomous driving". In: *Web Intelligence*. Vol. 22. 2. SAGE Publications Sage UK: London, England. 2024, pp. 185–207.

[23] Chenguang Li, Boheng Zhang, Jia Shi, and Guangliang Cheng. "Multi-level domain adaptation for lane detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4380–4389.

[24] Yongqi Dong, Sandeep Patil, Bart Van Arem, and Haneen Farah. "A hybrid spatial–temporal deep learning architecture for lane detection". In: *Computer-Aided Civil and Infrastructure Engineering* 38.1 (2023), pp. 67–86.

[25] Jiao Zhan, Jingnan Liu, Yejun Wu, and Chi Guo. "Multi-task visual perception for object detection and semantic segmentation in intelligent driving". In: *Remote Sensing* 16.10 (2024), p. 1774.

[26] Seungwoo Yoo, Hee Seok Lee, Heesoo Myeong, Sungrack Yun, Hyoungwoo Park, Janghoon Cho, and Duck Hoon Kim. "End-to-end lane marker detection via row-wise classification". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 1006–1007.

[27] Xinyu Zhang, Yan Gong, Jianli Lu, Zhiwei Li, Shixiang Li, Shu Wang, Wenzhuo Liu, Li Wang, and Jun Li. "Oblique convolution: A novel convolution idea for redefining lane detection". In: *IEEE Transactions on Intelligent Vehicles* 9.2 (2023), pp. 4025–4039.

[28] Zhuoling Li, Chunrui Han, Zheng Ge, Jinrong Yang, En Yu, Haoqian Wang, Xiangyu Zhang, and Hengshuang Zhao. "Grouplane: End-to-end 3d lane detection with channel-wise grouping". In: *IEEE Robotics and Automation Letters* (2024).

[29] Zhongyu Yang, Chen Shen, Wei Shao, Tengfei Xing, Runbo Hu, Pengfei Xu, Hua Chai, and Ruini Xue. "LDTR: Transformer-based lane detection with anchor-chain representation". In: *Computational Visual Media* 10.4 (2024), pp. 753–769.

[30] Shaofei Huang, Zhenwei Shen, Zehao Huang, Zi-han Ding, Jiao Dai, Jizhong Han, Naiyan Wang, and Si Liu. "Anchor3dlane: Learning to regress 3d anchors for monocular 3d lane detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17451–17460.

[31] Zequn Qin, Pengyi Zhang, and Xi Li. "Ultra Fast Deep Lane Detection With Hybrid Anchor Driven Ordinal Classification". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 46.5 (May 2024), pp. 2555–2568. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2022.3182097. URL: https://doi.org/10.1109/TPAMI.2022.3182097.

[32] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. "Polylanenet: Lane estimation via deep polynomial regression". In: *2020 25th international conference on pattern recognition (ICPR)*. IEEE. 2021, pp. 6150–6156.

[33] Ruijin Liu, Zejian Yuan, Tie Liu, and Zhiliang Xiong. "End-to-end lane shape prediction with transformers". In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2021, pp. 3694–3702.

[34] Yunqian Fan, Xiuying Wei, Ruihao Gong, Yuqing Ma, Xiangguo Zhang, Qi Zhang, and Xianglong Liu. "Selective focus: investigating semantics sensitivity in post-training quantization for lane detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 11. 2024, pp. 11936–11943.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *arXiv:1512.03385 [cs]* (Dec. 10, 2015). arXiv: 1512.03385. (Visited on 04/04/2022).

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature Pyramid Networks for Object Detection". In: *arXiv:1612.03144 [cs]* (Apr. 19, 2017). arXiv: 1612.03144. (Visited on 04/04/2022).

[37] Gang Li, Di Xu, Xing Cheng, Lingyu Si, and Changwen Zheng. "SimViT: Exploring a Simple Vision Transformer with sliding windows". In: *arXiv:2112.13085 [cs]* (Dec. 24, 2021). arXiv: 2112.13085. (Visited on 04/04/2022).

[38] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (Feb. 1, 2020), pp. 318–327. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2018.2858826. (Visited on 04/04/2022).

[39] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. "Dice Loss for Data-imbalanced NLP Tasks". In: *arXiv:1911.02855 [cs]* (Aug. 29, 2020). arXiv: 1911.02855. (Visited on 04/04/2022).

[40] Tusimple. *Tusimple lane detection benchmark*. 2017.

[41] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling". In: *CoRR* abs/1805.04687 (2018). arXiv: 1805.04687. URL: http://arxiv.org/abs/1805.04687.

[42] Lingyun Han, Kun Xu, Wensheng Hu, and Zhanwen Liu. "Lane Detection Method Based on MCA-UFLD". In: *2023 IEEE 8th International Conference on Intelligent Transportation Engineering (ICITE)*. IEEE. 2023, pp. 146–152.

[43] Yufeng Du, Rongyun Zhang, Peicheng Shi, Linfeng Zhao, Bin Zhang, and Yaming Liu. "ST-LaneNet: lane line detection method based on swin transformer and LaneNet". In: *Chinese Journal of Mechanical Engineering* 37.1 (2024), p. 14.

[44] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change 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.

[45] Yeongmin Ko, Younkwan Lee, Shoaib Azam, Farzeen Munir, Moongu Jeon, and Witold Pedrycz. "Key points estimation and point instance segmentation approach for lane detection". In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2021), pp. 8949–8958.

[46] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. "Resa: Recurrent feature-shift aggregator for lane detection". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 4. 2021, pp. 3547–3554.

[47] Xu Cao, Weisheng Liu, and Zhijian Wang. "Adaptive ROI Optimization Pyramid Network: Lane Detection for FSD under Data Uncertainty." In: *Engineering Letters* 33.2 (2025).

[48] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. *Deep Layer Aggregation*. Jan. 4, 2019. arXiv: 1707.06484[cs]. URL: http://arxiv.org/abs/1707.06484 (visited on 09/05/2022).

[49] Jonah Philion. "Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 11582–11591.

[50] Dan Zhang, Guolv Zhu, Shibo Lu, and Chang Li. "Lane Detection Based on Improved RESA in Power Plant". In: *2024 IEEE 4th International Conference on Power, Electronics and Computer Applications (ICPECA)*. IEEE. 2024, pp. 108–112.

[51] Zhan Qu, Huan Jin, Yang Zhou, Zhen Yang, and Wei Zhang. "Focus on local: Detecting lane marker from bottom up via key point". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 14122–14130.

[52] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. "Condlanenet: a top-to-down lane detection framework based on conditional convolution". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 3773–3782.

[53] Jing Zhao, Zengyu Qiu, Huiqin Hu, and Shiliang Sun. "HWLane: HW-transformer for lane detection". In: *IEEE Transactions on Intelligent Transportation Systems* (2024).

[54] Shuyan Wang, Ya Liu, and Feng Zhang. "A Multi-Task Autonomous Driving Environment Perception Network Based on CA-YOLOPv8". In: *2024 20th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE. 2024, pp. 1–9.

[55] Dat Vu, Bao Ngo, and Hung Phan. "HybridNets: End-to-End Perception Network". In: *arXiv e-prints* (2022), arXiv–2203.

AUTHORS

Mian Zhou received a Doctor's degree from the Department of Computer Science, University of Reading, UK. He is currently a senior associate professor at the School of AI and Advanced Computing, XJTLU Entrepreneur College (Taicang), Xi'an Jiaotong-Liverpool University. His research interests include computer vision, image processing, and pattern recognition.

Guoqiang Zhu is a graduate student at Tianjin University of Technology, specializing in computer vision and machine learning. His research focuses on leveraging deep learning techniques for image recognition.

ZHIKUN FENG is pursuing a PhD at the University of Electronic Science and Technology of China. His research interests include machine learning and computer vision.

HAOYI LIAN is an undergraduate student at the School of AI and Advanced Computing, XJTLU Entrepreneur College (Taicang), Xi'an Jiaotong-Liverpool University. She is majoring in data science and big data technology and is particularly interested in the fields of machine learning, deep learning and computer vision.

SIQI HUANG is an assistant professor in the School of AI and Advanced Computing at XJTLU Entrepreneur College (Taicang). He received his Ph.D. in electrical engineering at The University of North Carolina at Charlotte in 2022, and BEng in software engineering from Sun Yat-sen University in 2015.His research interests include AI-driven video streaming system optimization; energy and latency analysis and optimization of AI (DNN) models on mobile devices (TinyML); real-time HD map generation and updates for autonomous driving and software OTA update services for autonomous vehicles; mobile edge computing with embedded AI devices; mobile AR/VR and human-computer interaction system.