Fast and Accurate Lane Detection via Frequency Domain Learning

Yulin He
National University of Defense Technology
heyulin@nudt.edu.cn

Wei Chen*
National University of Defense Technology
chenwei@nudt.edu.cn

Zhengfa Liang
Defense Innovation Institute
liangzhengfa10@nudt.edu.cn

Dan Chen
Wuhan University
dan.chen@whu.edu.cn

Yusong Tan
National University of Defense Technology
ystan@nudt.edu.cn

Xin Luo
National University of Defense Technology
luoxin13@nudt.edu.cn

Chen Li
National University of Defense Technology
lichen14@nudt.edu.cn

Yulan Guo
National University of Defense Technology
yulan.guo@nudt.edu.cn

ABSTRACT

It is desirable to maintain both high accuracy and runtime efficiency in lane detection. State-of-the-art methods mainly address the efficiency problem by direct compression of high-dimensional features. These methods usually suffer from information loss and cannot achieve satisfactory accuracy performance. To ensure the diversity of features and subsequently maintain information as much as possible, we introduce multi-frequency analysis into lane detection. Specifically, we propose a multi-spectral feature compressor (MSFC) based on two-dimensional (2D) discrete cosine transform (DCT) to compress features while preserving diversity information. We group features and associate each group with an individual frequency component, which incurs only 1/7 overhead of one-dimensional convolution operation but preserves more information. Moreover, to further enhance the discriminability of features, we design a multi-spectral lane feature aggregator (MSFA) based on one-dimensional (1D) DCT to aggregate features from each lane according to their corresponding frequency components. The proposed method outperforms the state-of-the-art methods (including LaneATT and UFLD) on TuSimple, CULane, and LLAMAS benchmarks. For example, our method achieves 76.32% F1 at 237 FPS and 76.98% F1 at 164 FPS on CULane, which is 1.23% and 0.30% higher than LaneATT. Our code and models are available at https://github.com/harrylin-hyl/MSLD.

*Wei Chen is the corresponding author. Yulin He and Wei Chen are co-first authors.

MM '21, October 20–24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475267

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Networks** → *Network algorithms.*

KEYWORDS

lane detection; multi-frequency analysis; feature diversity; discrete cosine transform; frequency domain learning

ACM Reference Format:
Yulin He, Wei Chen, Zhengfa Liang, Dan Chen, Yusong Tan, Xin Luo, Chen Li, and Yulan Guo. 2021. Fast and Accurate Lane Detection via Frequency Domain Learning. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475267

1 INTRODUCTION

Lane detection is an essential perception task in computer vision, which has long been placed in the core of autonomous driving systems. Although it has been investigated for a long time, it is still challenging to achieve good performance in conditions with occlusions, bad weather, and complex road scenes. The inherent long and thin properties of the lane itself further increases the difficulty. Besides, as a fundamental component of autonomous driving, the lane detection algorithm is heavily executed, which requires a strong real-time efficiency to save processing power for other systems. It is desirable but very challenging to maintain both high accuracy and runtime efficiency in lane detection.

Most algorithms [10, 21, 33] typically treat lane detection as a semantic segmentation problem, i.e., localizing lanes by pixel-wise classification with an auto-encoder structure. The lanes are represented as segmented binary features rather than lines or curves in segmentation methods, which makes it difficult to explicitly utilize the slender shape prior information of lanes. In addition, due to the long-distance crossing of lanes, the segmented features require more powerful long-distance perception capabilities than other generic semantic segmentation tasks. Some methods [21, 33] attempt to pass spatial information within feature maps by

defining special convolutions. Nevertheless, these operations are time-consuming. Besides, a clustering post-processing step is also required to separate predictions to different lanes, which further hinders real-time efficiency.

Several solutions [13, 26, 32] have been proposed to solve the aforementioned problems. Most of these methods discard the heavy decoder network and directly exploit the high-dimensional features generated from the encoder network. To achieve a fast running time, one-dimensional convolution is usually as the de-facto standard choice to compress features [23, 27]. However, due to the simple structure of one-dimensional convolution and the big dimension gap between feature maps (from 512 to 8 or 64 [23, 27]), excessive information is lost. Reducing the dimension of features is necessary to improve the efficiency, while maintaining diverse information is essential to achieve high accuracy performance.

Channel attention network [11, 31] is an effective method to enhance the diversity of features by performing different attentions in the channel dimension. SENet [11] is one of the effective and well-known methods, which learns the channel weights through an additional network branch with global average pooling (GAP). Due to its simplicity and efficiency, GAP becomes the de-facto standard choice in the deep learning community. However, GAP is incapable of well capturing the rich input pattern information, and thus lacks feature diversity when processing different inputs. GAP is proved to be equivalent to the lowest frequency component of DCT [24]. Therefore, only using GAP will discard the other frequency components, which contains much useful information on feature channels.

In this paper, we introduce multi-frequency learning to preserve information as much as possible when compressing features. Specifically, we propose a multi-spectral feature compressor (MSFC) based on 2D DCT, which exploits different frequency components by grouping features and associating each of them with an individual frequency component. By incorporating more frequency components into feature compression processing, the information from these different frequency components can be exploited. The multi-spectral description enhances the diversity of features. As a result, the compressed features can preserve more information.

Besides, the long and thin properties of the lane makes it difficult to aggregate features. Although anchor-based methods can easily extract the lane feature according to the positions of the predefined anchor lines, the information between multiple pixels of each lane is not fully communicated due to the distance, thereby losing discrimination of the lane feature. Hence, we design a multi-spectral lane feature aggregator (MSFA) based on 1D DCT, which handles the pixels from a new feature dimension and correlates pixels with weighted sums of a variety of frequencies. DCT can easily encode the relative positions of pixels, and the weighted summation of different frequencies can also enhance the connection between pixels.

Extensive experiments are conducted on TuSimple [28], CULane [21], and LLAMAS [1]. Our multi-spectral lane detector (MSLD) achieves higher efficacy and efficiency compared with current state-of-the-art methods. Moreover, our MSFC can be directly applied to existing methods. We achieve 1.18 % F1 score gain without speed reduction when applying MSFC to UFLD [23] (an ultra-fast lane detector), which shows the generalization ability of our method.

In summary, our main contributions are as follows:

- We propose a novel and general technique (i.e., MSFC) to pursuit faster speed while maintaining high accuracy for lane detection. MSFC compresses features using multiple frequency descriptions of DCT in feature channels, while preserving information as much as possible.
- We develop an effective and efficient lane feature aggregator (i.e., MSFA), which enhances the information communication between pixels in each lane by weighted sums of a variety of frequencies.
- We introduce a fast and accurate lane detector (MSLD) based on anchor lines and multi-frequency analysis. Our detector achieves the state-of-the-art performance on the TuSimple, CULane, and LLAMAS benchmarks.

2 RELATED WORK

2.1 Lane Detection

Lane detection is conventionally developed to obtain exact positions of lanes using hand-crafted features. These approaches [3, 19] require a sophisticated feature engineering process, and are only applicable in relatively simple driving scenarios, hindering their applications in complicated real-world scenarios.

Deep learning based methods avoid manual feature design by extracting features in an end-to-end manner. In earlier works [10, 21], lane detection is usually formulated as a segmentation task. These methods take an image as their input and produce a segmentation map with pixel-wise predictions. However, their performance is usually limited due to the difficulties on learning such long and thin structures. To address this issue, specialized feature aggregators are designed for lane detection [21, 33], which exploit visual cues and spatial information by message passing. However, their speed is slow, which hinders their applicability in real-world cases. To push the efficiency of algorithms into real-time, some methods utilize knowledge distillation [9, 10] or neural architecture search [14] to learn a smaller backbone with considerable accuracy. Nevertheless, the time-consuming feature aggregation processing and heavy decoder network are still the obstacle to improving the speed of these segmentation-based methods.

Recently, some works attempt to apply different modeling manners for lane detection, including row-based classification [23, 32], polynomial-based [16, 26], and anchor-based methods [2, 13, 27]. In row-based classification methods, lanes are detected by row-wise classification based on a grid division of the input image, which is first introduced in E2E-LMD [32]. In UFLD [23], the capability of reaching high speed by feature compression (from 512 to 8) is demonstrated, although some accuracy is lost. Polynomial-based methods (e.g., PolyLaneNet [26]) consider lanes as polynomial curves and detect lanes by deep polynomial regression. Later on, LSTR [16] achieves the state-of-the-art results on the TuSimple benchmark by exploiting the transformer network [29] and the Hungarian algorithm. Anchor-based approaches generate a large number of anchor points or anchor lines in images and detect lanes by classification and regression. In PointLaneNet [2], the lane location is extracted by predicting the offsets from each anchor point. In Line-CNN [13], a line proposal unit is proposed to generate a set of rays to capture the actual lane. In LaneATT [27], an effective

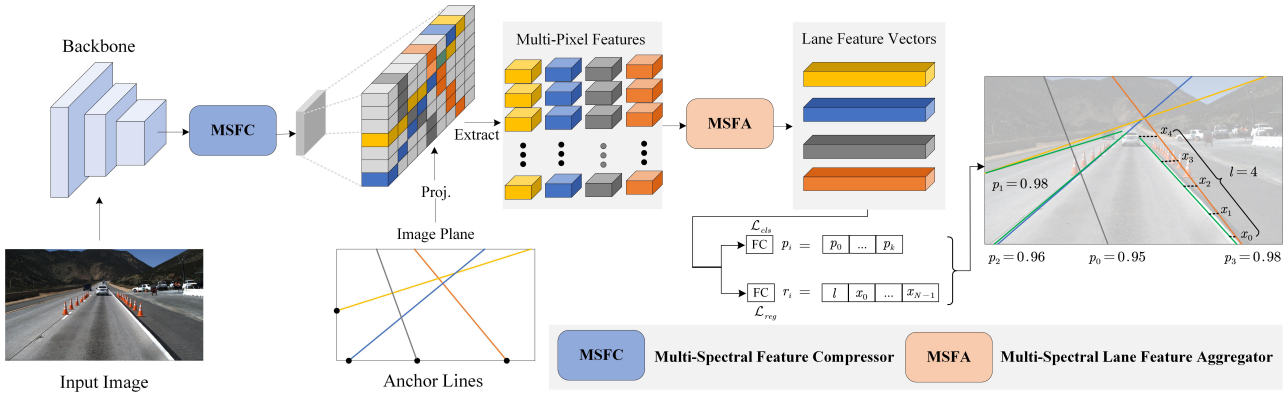


Figure 1: An overview of our method. Backbone generates high-dimensional feature maps from an input image. MSFC is then applied to reduce the dimension of features. Subsequently, a series of predefined anchor lines are projected onto the feature maps, and multi-pixel features can be obtained by pixel-wise extraction. Then, MSFA aggregates multi-pixel features to lane feature vectors. Finally, the features vectors are fed into two layers (one for classification and another for regression) to make the final predictions.

attention mechanism is proposed to aggregate global features for lanes. Beside, features (from 512 to 64) are compressed for high speed. These aforementioned methods do not take the information loss in feature compression into account, and do not consider the information communication between pixels in each lane.

2.2 Frequency Domain Learning

Frequency analysis is a powerful tool in signal processing. Recently, frequency analysis has been introduced to the area of deep learning. In [4, 6], frequency analysis is introduced into CNNs for JPEG encoding. For efficient network training, [5, 7, 20, 30] focus on network compression via kernel pruning, learned quantization, and entropy encoding. Then, DCT is incorporated in [24] to channel attention networks, capturing rich input pattern information. This paper further introduces frequency analysis into lane detection and provides a novel way to maintain accuracy as much as possible while achieving high speed.

3 METHOD

In this section, we first introduce the architecture of our MSLD. Then, we revisit the DCT frequency analysis, and introduce our multi-spectral feature compressor (MSFC) and multi-spectral lane feature aggregator (MSFA) based on DCT. Finally, we elaborate on the details of the model training process.

3.1 Architecture

MSLD is a single-stage anchor-based model (similar to YOLOv3 [25] and SSD [18]) for lane detection. An overview of our method is shown in Fig. 1. It takes an RGB image $I \in \mathbb{R}^{H_I \times W_I \times 3}$ as its input, which is acquired from a front-facing camera mounted on a vehicle. Then, a backbone network (such as ResNet [8]) is used to extract the features from I and produce high-dimensional feature maps $F_H \in \mathbb{R}^{H_F \times W_F \times C}$ with deep semantic information. To reduce calculation and model parameters, MSFC is applied to F_H, resulting in compressed features $F_C \in \mathbb{R}^{H_F \times W_F \times \hat{C}}$ with a low dimension. MSFC exploits multi-spectral description to improve features' diversity, thereby retaining more information than one-dimensional convolution. Anchor lines are then generated (N_{anc} in total) in the image plane. Each anchor line is defined as a ray by a start coordinate $\{X_s, Y_s\}$ (located at the left, bottom and right borders of the image) and its slope θ. Subsequently, anchor lines are projected from the image plane onto the feature space. To ensure that the dimension of the feature of each anchor line is the same, we uniformly sample pixels in the height dimension of the feature map $y_i = \{0, 1, 2, ..., H_F - 1\}$. The corresponding x-coordinates can be obtained by a projection function:

$$x_i = \left\lfloor \frac{1}{\tan\theta}(y_i - Y_s/S) + X_s/S \right\rfloor \tag{1}$$

where S is the global stride of the backbone. Then, we can extract multi-pixel features $F_P = \{F_P^1, F_P^2, ..., F_P^{H_F-1}\}$ for each anchor line based on projected coordinates. In cases where $\{x_i, y_i\}$ is outside the boundaries of F_C, F_P^i is zero-padded. To enhance the communication between multiple pixels in each lane, MSFA is then applied onto F_P, aggregating lane feature vectors F_L. Finally, the predictions are obtained by classification and regression subnets, which are implemented as fully connected layers (FC). Since the lane is represented by 2D-points with fixed equally-spaced y-coordinates, the corresponding x-coordinates and the length of the lane are regressed using the regression subnet.

3.2 Revisiting Discrete Cosine Transform

A discrete cosine transform (DCT) characterizes a finite sequence of data points as a sum of cosine functions oscillating at different frequencies. Typically, the definition of 1D DCT can be written as:

$$F(k) = c(k) \sum_{i=0}^{L-1} x_i \underbrace{\cos\left(\frac{\pi k}{L}(i + \frac{1}{2})\right)}_{\text{DCT weights}} \tag{2}$$

$$c(k) = \begin{cases} \sqrt{\frac{1}{L}} & k = 0 \\ \sqrt{\frac{2}{L}} & k \neq 0 \end{cases} \tag{3}$$

where $F(k) \in \mathbb{R}^L$ is the k-th frequency spectrum of DCT, $x \in \mathbb{R}^L$ is the input, and L is the length of the input x. Similarly, 2D DCT can be written as:

$$F(h,w) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} x_{ij}^{2d} \underbrace{\cos(\frac{\pi h}{H}(i+\frac{1}{2}))\cos(\frac{\pi w}{W}(j+\frac{1}{2}))}_{\text{DCT weights}} \tag{4}$$

where $F(h,w) \in \mathbb{R}^{H \times W}$ is the 2D frequency spectrum at (h,w) index, $x^{2d} \in \mathbb{R}^{H \times W}$ is the input, H and W are the height and width of x^{2d}, respectively. The corresponding inverse 2D DCT can be written as:

$$x_{i,j}^{2d} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} F(h,w)\cos(\frac{\pi h}{H}(i+\frac{1}{2}))\cos(\frac{\pi w}{W}(j+\frac{1}{2})). \tag{5}$$

Note that, we ignore some constant normalization factors ($c(k)$) for simplicity in Eq. 4 and Eq. 5, which do not affect the results in this work. DCT can be viewed as a weighted sum of inputs with the cosine parts (DCT weights in Eq. 2 and Eq. 4).

Global average pooling (GAP) is an operation of averaging in the spatial dimension, and it can be considered as the lowest frequency component of input. Derived from Eq. 4, we have:

$$\begin{aligned} F(0,0) &= \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} x_{i,j}^{2d} \cos(\frac{0}{H}(i+\frac{1}{2}))\cos(\frac{0}{W}(j+\frac{1}{2})) \\ &= \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} x_{i,j}^{2d} \\ &= GAP(x^{2d})HW \end{aligned} \tag{6}$$

This means that using GAP to quantify feature channels is equivalent to discarding the useful information of other frequency components. Therefore, the use of multi-frequency analysis can preserve more frequency information, thereby enhancing feature diversity.

3.3 Multi-Spectral Feature Compressor

In this section, we introduce the module design of MSFC, then analyze the complexity of one-dimensional convolution and MSFC.

Module design of MSFC. MSFC is a two-step feature compressor to reduce feature dimensions while retaining rich information. It includes two parts: multi-spectral quantification and group compression, as shown in Fig. 2.

During multi-spectral quantification, we first split the input $F_H \in \mathbb{R}^{H_F \times W_F \times C}$ into n parts along the channel dimension. Let $[F_H^0, F_H^1, ..., F_H^{n-1}]$ be the parts, where $F_H^i \in \mathbb{R}^{H_F \times W_F \times C'}$, $C' = \frac{C}{n}$, and C should be divisible by n. Each part is assigned with a corresponding 2D DCT frequency component ranging from low frequency to high frequency, and the 2D DCT results can be computed as:

$$\begin{aligned} Freq^i &= \text{DCT}_{2D}(u,v)(F_H^i) \\ &= \sum_{h=0}^{H_F-1} \sum_{w=0}^{W_F-1} \text{w}_{h,w}^{u,v} x_{h,w}^i \end{aligned} \tag{7}$$

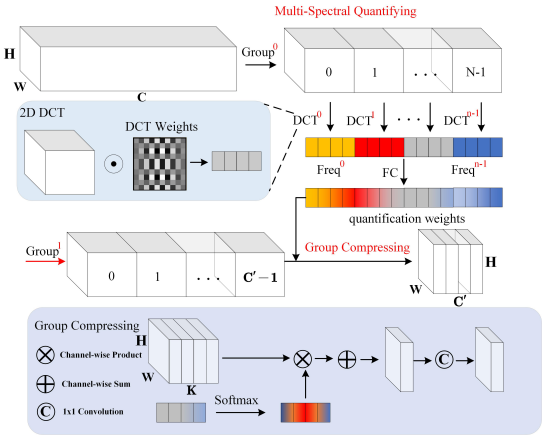


Figure 2: Illustration of MSFC. For simplicity, the 2D DCT indices are represented in the 1D format. The red arrow indicates F_H, which is the input of MSFC. Best viewed in colors.

where $Freq^i \in \mathbb{R}^{C'}$ is a C'-dimensional vector in the i-th frequency, (u,v) are the frequency component's 2D index corresponding to F_H^i, $\text{w}_{h,w}^{u,v}$ is the DCT weight in Eq. 4, and $x_{h,w}^i$ is the corresponding value of F_H^i. Multi-frequency vector $Freq \in \mathbb{R}^C$ can be obtained by concatenation:

$$Freq = \text{cat}([Freq^0, Freq^1, \cdots, Freq^{n-1}]). \tag{8}$$

Once multi-spectral descriptions $Freq$ are obtained, quantification weights $W_Q \in \mathbb{R}^C$ are learned by two fully connected layers: $W_Q = fc(Freq)$.

During group compression, another division is applied to split F_H and W_Q into $\hat{C}$ groups, and C should be divisible by $\hat{C}$. Let $[F_H^0, F_H^1, ..., F_H^{\hat{C}-1}]$ and $[W_Q^0, W_Q^1, ..., W_Q^{\hat{C}-1}]$ be the groups, each of the groups has $K = \frac{C}{\hat{C}}$ channels, and the compressed feature $F_C \in \mathbb{R}^{H_F \times W_F \times \hat{C}}$ of the i-th group can be written as:

$$F_C^i = Conv(\sum_{k=0}^{K} F_H^i(k) \odot softmax(W_Q^i(k))) \tag{9}$$

where $softmax$ function is applied to transform each group's quantification weights W_Q^i into soft weights, element-wise weighted summation is used to reduce the dimension from K to 1 in each group, and an one-dimensional convolution is utilized for low-dimensional feature integration.

Complexity analysis. We analyze the computational complexity of our MSFC and compare it with one-dimensional convolution.

The computational complexity of one-dimensional convolution is written as:

$$O_{Conv} = 1 \times 1 \times H_F \times W_F \times C \times \hat{C} \tag{10}$$

while the computational complexity of MSFC is written as:

$$O_{MSFC} = 2 \times C \times \frac{C}{R} + H_F \times W_F \times \hat{C} \times \hat{C} \tag{11}$$

where R is the compression ratio in two fully connected layers. The first one reduces the dimension to $\frac{C}{R}$ while the second one reverts the dimension to C for reducing complexity. Taking $H_F = 12$,

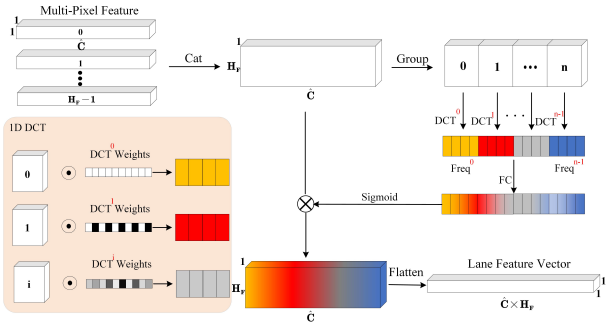


Figure 3: Illustration of MSFA. Best viewed in colors.

$W_F = 20$, $C = 512$, $R = 16$, and $\hat{C} = 64$ for example, the computational complexity of our method is about 1/7 of one-dimensional convolution according to Eq. 10 and Eq. 11. Our method has a lower computational complexity, but can better preserve the information in feature compression, thereby improving the performance.

3.4 Multi-Spectral Feature Aggregator

Due to the problems such as occlusion and illumination changes in lane detection, long-distance communication is extremely important, which enhances the capability of a network to capture visual cues. However, the local perceptual properties of convolution make it unable to establish a long-distance connection between multiple pixels of the lane, resulting in reduced discriminability of the lane feature. To address this problem, we propose MSFA (as shown in Fig. 3), which exploits the different weighted sums in DCT to perform multi-frequency communication between pixels of the lane.

In details, MSFA first concatenates multi-pixel features in another feature dimension $F_P \in \mathbb{R}^{H_F \times \hat{C}}$, and then splits F_P into n parts: $[F_P^0, F_P^1, ..., F_P^{n-1}]$. The feature of each part can be transformed to its corresponding frequency component by 1D DCT:

$$\begin{aligned} Freq^i &= \mathrm{DCT}_{1\mathrm{D}}(k)(F_P^i) \\ &= \sum_{l=0}^{H_F-1} \mathrm{W}_l^k \, x_l^i \end{aligned} \tag{12}$$

where $k \in \{0, 1, 2, ..., n\}$ is the 1D index of a frequency component corresponding to F_P^i, W_l^k is the DCT weight in Eq. 2, and x_l^i is the corresponding value of F_P^i. Then, multiple frequencies $Freq \in \mathbb{R}^{\hat{C}}$ can be obtained by concatenation:

$$Freq = \mathrm{cat}([Freq^0, Freq^1, \cdots, Freq^{n-1}]). \tag{13}$$

The whole calculation can be written as:

$$F_L = Flatten(F_P \odot sigmoid(fc(Freq))) \tag{14}$$

where $F_L \in \mathbb{R}^{1\times1\times\hat{C}H_F}$ is the lane feature vector of the anchor line, fully connected layers (fc) and *sigmoid* function are applied to learn the importance of channels, element-wise product is used to perform different concerns, and *Flatten* operation is used to reshape the features to a vector.

3.5 Model Training

Training sample assignment is an essential step in model training. For two lanes with common valid indices (y-coordinates), the x-coordinates are $X_a = \{x_i^a\}_{i=1}^{N_{pts}}$ and $X_b = \{x_i^b\}_{i=1}^{N_{pts}}$, respectively, where N_{pts} is the number of evaluation points. The lane distance metric proposed in [13] is adopted to compute the distance between two lanes:

$$D(X_a, X_b) = \begin{cases} \frac{1}{e'-s'+1} \cdot \sum_{i=s'}^{e'} \left| x_i^a - x_i^b \right|, & e' \geq s' \\ +\infty, & \text{else} \end{cases} \tag{15}$$

where s_a and s_b are the start valid indices of two lanes, e_a and e_b are the end valid indices of two lanes, $s' = max(s_a, s_b)$ and $e' = min(e_a, e_b)$ define the range of those common indices. Then, calculating the distance between anchor lines and target lanes, the anchor lines with distance lower than a threshold τ_p are considered as positive samples, while those with distance larger than a threshold τ_n are considered as negative samples.

The final loss is composed of two terms, i.e., classification loss and regression loss, which are implemented by Focal loss [15] and L1 loss, respectively. The final loss is defined as:

$$\begin{aligned} \mathcal{L}\left(\{\mathbf{p}_i, \mathbf{r}_i\}_{i=0}^{N_{p\&n}-1}\right) =& \lambda \sum_i \mathcal{L}_{cls}\left(\mathbf{p}_i, \mathbf{p}_i^*\right) \\ &+ \sum_i \mathcal{L}_{reg}\left(\mathbf{r}_i, \mathbf{r}_i^*\right) \end{aligned} \tag{16}$$

where $N_{p\&n}$ is the number of positive and negative samples, λ is used to balance the loss terms, $\mathbf{p}_i$, $\mathbf{r}_i$ are the classification and regression predictions of the i-th anchor line, and $\mathbf{p}_i^*$, $\mathbf{r}_i^*$ are the corresponding classification and regression targets. $\mathbf{p}_i^*$ consists of "0" and "1", i.e., background and lanes. $\mathbf{r}_i^*$ is composed with the length l and the x-coordinates.

4 EXPERIMENTS

Datasets. We conduct experiments on three datasets, including two widely used lane detection datasets (TuSimple [28] and CULane [21]) and a recently released dataset (LLAMAS [1]). The TuSimple dataset is collected with stable lighting conditions in highways. The CULane dataset consists of nine different scenarios, including normal, crowd, dazzle, shadow, no line, arrow, curve, cross, and night in urban areas. LLAMAS is a large lane detection dataset in highways, which is generated by high definition maps without manual annotation. More details about these datasets can be seen in Table 1.

Evaluation metrics.

1) TuSimple. The evaluation metric in the TuSimple dataset is accuracy. Accuracy is calculated as $accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}}$, where C_{clip} is the number of lane points predicted correctly and S_{clip} is the total number of evaluation points in each clip. For each evaluation point, if the predicted point and the target point are within 20 pixel values, the prediction is considered to be correct, otherwise wrong. Besides, we also calculate the rate of false-positive (FP), the rate of false-negative (FN), and F1 score on predictions.

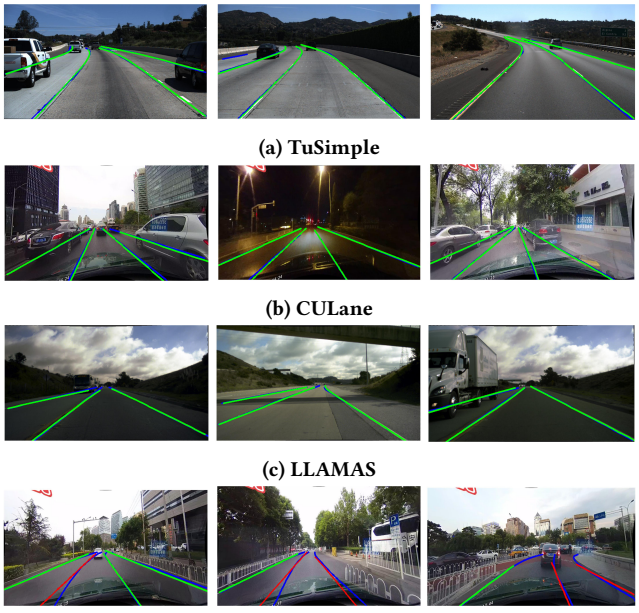
2) CULane and LLAMAS. The evaluation metric on CULane and LLAMAS is F1 score. Different from TuSimple, each lane is considered as a line with a width of 30 pixels. Intersection-over-union

Table 1: Dataset description.

Dataset	#Frame	Train	Validation	Test	Resolution	#Lane	#Scenarios	environment
TuSimple [28]	6,408	3,268	358	2,782	1280×720	≤ 5	1	highway
CULane [21]	133,235	88,880	9,675	34,680	1640×590	≤ 4	9	urban and highway
LLAMAS [1]	100,042	58,269	20,844	20,929	1276×717	≤ 4	1	highway

Table 2: State-of-the-art results on TuSimple. For fair comparison, FPS was measured on the same machine used by our method. The best and second-best results across methods are shown in boldface and underlined, respectively.

Method	F1 (%)	Acc(%)	FP(%)	FN(%)	FPS	MACs(G)
FastDraw (ResNet-18) [22]	94.59	94.90	6.10	4.70	-	-
LSTR (Simple-ResNet) [16]	**96.86**	96.18	**2.91**	3.38	120.0	**0.574**
Line-CNN [13]	96.79	**96.87**	4.42	<u>1.97</u>	30.0	-
PointLaneNet [2]	95.07	96.34	4.67	5.18	71.0	-
E2E-LMD (ResNet-18) [32]	96.40	96.04	<u>3.11</u>	4.09	-	-
SCNN [21]	95.97	96.53	6.17	**1.80**	7.5	-
ENet-SAD [10]	95.92	<u>96.64</u>	6.02	2.05	75.0	-
UFLD (ResNet-18) [23]	87.87	95.82	19.05	3.92	**425.0**	-
PolyLaneNet [26]	90.62	93.36	9.42	9.33	115.0	<u>1.7</u>
LaneATT (ResNet-18) [27]	96.71	95.57	3.56	3.01	<u>250.0</u>	9.3
LaneATT (ResNet-34) [27]	96.77	95.63	3.53	2.92	171.0	18.0
MSLD (ResNet-18)	96.77	95.62	3.37	3.07	237.0	9.4
MSLD (ResNet-34)	<u>96.84</u>	95.73	3.26	3.05	164.0	18.1



(a) TuSimple

(b) CULane

(c) LLAMAS

(d) Failure Cases

Figure 4: Visualization results on TuSimple (top row), CULane (second row), and LLAMAS (third row). Some failure cases on CULane are shown in the bottom row. Blue lines are ground-truth, while green and red lines are true-positives and false-positives, respectively.

(IoU) is calculated between predictions and targets. Those predictions with IoUs larger than a threshold (e.g., 0.5) are considered as correct. F1 score is defined as: $F_1 = \frac{2\times Precision\times Recall}{Precision+Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

Efficiency metrics. Two efficiency-related metrics are reported, i.e., frames-per-second (FPS) and multiply-accumulate operations (MACs). One MAC is approximately equal to two floating operations (FLOPs), i.e., one multiplication and one addition. FPS is computed using a single image per batch and constant inputs, so the metric is not dependent on I/O operations but only the model's efficiency.

Implementation details. To save memory usage, the original images of all datasets are resized to 360×640. For data augmentation, a random affine transformation is performed (with translation, rotation, and scaling) along with random horizontal flips. Adam [12] is adopted as the optimizer to train our model, and the learning rate is set to 0.003 with the *CosineAnnealingLR* learning rate schedule. The total number of training epochs is set to 20 for CULane and LLAMAS, and 100 for TuSimple. All experiments and FPS measures are conducted on a machine with a RTX 2080 Ti GPU. The model parameters are set as: $\hat{C} = 64$, $N_{pts} = 72$, $N_{anc} = 1000$, $\tau_p = 15$, and $\tau_n = 20$. In MSFC, we adopt the 2D DCT weights in [24], with 16 frequency components in total. In MSFA, we adopt the 1D DCT weights, whose indices are from 0 to 7.

4.1 Comparison with State-of-the-Art

In this section, we show the results on three lane detection benchmarks, i.e., TuSimple, CUlane, and LLAMAS. In these experiments, ResNet-18 and ResNet-34 [8] are used as our backbone networks. The metrics for comparison include F1 score, running speed (FPS), and calculation amount (MACs).

For the TuSimple dataset, the results of MSLD and other state-of-the-art methods are shown in Table 2 and in Fig. 5 (left side). It can be clearly seen that the experimental results are relatively saturated, mainly due to the easy nature of scenes in this dataset and the relatively loose evaluation standards. Nevertheless, MSLD also achieves superior overall performance, especially in F1 score, FP ratio, and

Table 3: State-of-the-art results on CUlane. For fair comparison, F1 scores of multiple scenarios, runtime, and calculation amount were measured with a single batch size on the same machine and conditions. Because the images in "Cross" scene have no lanes, only false positives are shown. The best and second-best results across methods are shown in boldface and underlined, respectively.

Method	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	Cross	Night	FPS	MACs (G)
E2E-LMD (ResNet-18) [32]	70.80	90.00	69.70	60.20	62.50	43.20	83.20	**70.30**	2296	63.30	-	-
SCNN [21]	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10	7.5	-
ENet-SAD [10]	70.80	90.10	68.80	60.20	65.90	41.60	84.00	65.70	1998	66.00	75.0	-
RESA (ResNet-34) [33]	74.50	91.90	72.40	66.50	72.00	46.30	88.10	68.60	1896	69.80	-	-
ERFNet-IntRA-KD [9]	72.40	-	-	-	-	-	-	-	-	-	100.0	-
SIM-CycleGAN [17]	73.90	91.80	71.80	66.40	76.20	46.10	87.80	67.10	2346	69.40	-	-
CurveLanes-NAS-S [14]	71.40	88.30	68.60	63.20	68.00	47.90	82.50	66.00	2817	66.20	-	**9.0**
LaneATT (ResNet-18) [27]	75.09	91.11	72.96	65.72	70.91	48.35	85.49	63.37	1170	68.95	250.0	9.3
LaneATT (ResNet-34) [27]	76.68	92.14	**75.03**	66.47	78.15	49.39	**88.38**	67.72	1330	70.72	171.0	18.0
UFLD (ResNet-18) [23]	68.40	87.70	66.00	58.40	62.80	40.20	81.00	57.90	1743	62.10	425.0	-
UFLD + MSFC (ResNet-18)	69.58	89.10	67.69	56.84	60.08	39.23	82.83	55.78	1590	63.84	**427.5**	-
MSLD (ResNet-18)	76.32	91.68	74.39	67.77	71.54	49.95	87.22	65.53	**1074**	70.64	237.0	9.4
MSLD (ResNet-34)	**76.98**	**92.33**	74.73	**68.03**	**78.19**	**50.19**	88.04	67.99	1242	**72.15**	164.0	18.1

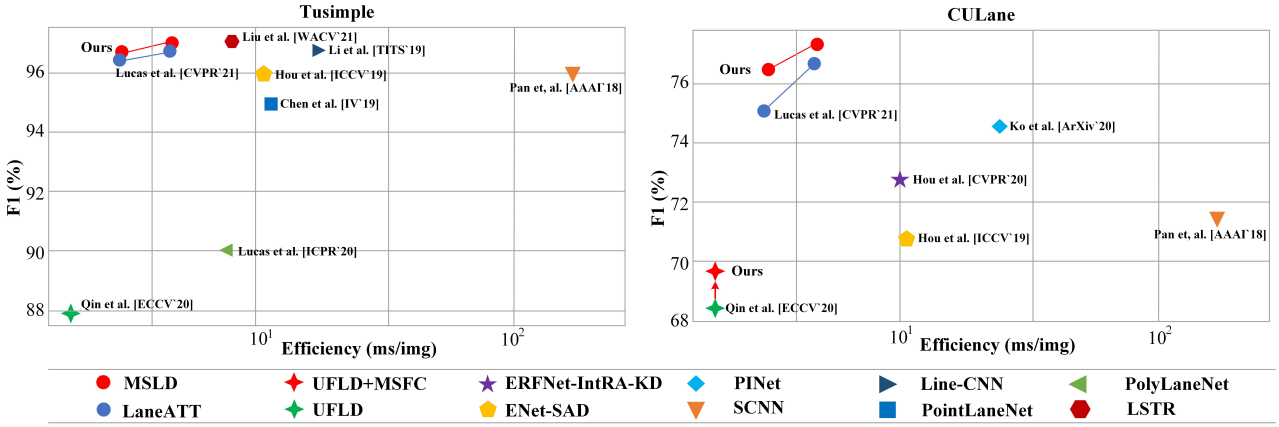


Figure 5: Running efficiency vs. F1 score of state-of-the-art methods on the TuSimple and CULane benchmarks.

Table 4: State-of-the-art results on LLAMAS. Prec. and Rec. denote precision and recall respectively.

Method	F1 (%)	Prec.(%)	Rec.(%)
PolyLaneNet [26]	88.40	88.87	87.93
LaneATT (ResNet-18) [27]	93.46	**96.92**	90.24
LaneATT (ResNet-34) [27]	93.74	96.79	**90.88**
MSLD (ResNet-18)	93.73	96.77	**90.88**
MSLD (ResNet-34)	**93.77**	96.88	90.85

FPS metrics. UFLD [23] is faster than ours, but its FP ratio is notably high (19.05%), which will easily cause false alarms, hindering its practical applications. LSTR [16] achieves higher accuracy with less computation cost than ours. However, its running speed is relative slow due to the time-consuming transformer network [29]. Besides, it does not perform well as in other complex datasets, such as CULane (about 64% F1 score produced by its code). Compared with LaneATT [27], our lightweight model (with ResNet-18) has comparable performance to their large model (ResNet-34) while being much faster (237 vs. 171 FPS on the same machine). Besides, our large model (ResNet-34) also achieves consistent performance improvements.

For the CULane dataset, the results of MSLD and other state-of-the-art methods are shown in Table 3 and in Fig. 5 (right side). Our MSLD achieves the state-of-the-art results while maintaining a fast running speed. Besides, the overall performance of MSLD in all scenarios is outstanding, which demonstrates the powerful generalization capability of our method to handle different scene changes. Compared with LaneATT [27] (whose performance is closest to ours), our MSLD improves performance in terms of F1 score, i.e., +1.23% and +0.30% with the ResNet-18 and ResNet-34 backbones while achieving almost the same speed, respectively. After introducing multi-spectrum analysis, there is a notable performance improvement with negligible additional calculations increase. This is a novel method to achieve high speed while maintaining high precision. Besides, when applying our proposed MSFC to UFLD [23], we achieve a F1 score gain of 1.18% with a higher speed (from

Table 5: Results of different grouping methods in MSFC on CULane with ResNet-18 backbone.

Method	Block	Random	Sort
F1 (%)	**75.76**	75.43	73.83

Table 6: Results of different numbers of 1D DCT weights in MSFA on CULane with ResNet-18 backbone.

n(DCT)	4	8	16
F1 (%)	76.04	**76.32**	75.18

425 FPS to 427.5 FPS). This also demonstrates that our method can consistently improve performance.

For the LLAMAS dataset, the results of MSLD, PolyLaneNet [26] and LaneATT [27] are shown in Table 4. Since LLAMAS is recently released, detection results of only two methods are provided. Compared with PolyLaneNet, the performance of MSLD is much better. Compared with LaneATT, our lightweight model (ResNet-18) achieves comparable performance to their large model (ResNet-34).

The visualization results of our method are shown in Fig. 4. Although the anchor lines are all straight in MSLD, it does not affect the fitting of curved lane lines, as shown in Fig. 4a. Besides, MSLD also performs well under various scenarios, such as crowded and night scenes, as shown in Fig. 4b. These results clearly demonstrate the effectiveness and generalization ability of our method. However, there are also some failure cases in the curve scene on CULane, as shown in Fig. 4d. We found that curved lanes only accounted for 1.2% of the training images on CULane. It results in significant deviations in training, thus affecting the predictions of curve lanes.

4.2 Ablation Study

Different grouping methods in MSFC. Grouping compression is the second step in MSFC, as shown in Fig. 2. To further investigate the influence of grouping methods in MSFC, we set up three grouping methods, including block grouping, random grouping, and sort grouping. These three methods divide high-dimensional features into several groups according to the original order, random order, and sort order, respectively. As shown in Table 5, the block grouping method achieves the best performance while the sort grouping method suffers from a large performance degradation. We guess that these various frequency features are independent and structural, so the disorder affects this property. Besides, sort grouping makes the training process unstable since it is exploited during training, which also affects its performance.

Different numbers of 1D DCT weights in MSFA. We investigate the effect of different 1D DCT weights numbers, i.e., 4, 8, and 16. These numbers should be divisible by $\hat{C}$ in order to divide channels into equal parts. As shown in Table 6, it works best when the number is 8 and worst when the number is 16. The number of sampling pixels for the anchor line feature is $H_F = 12$. We found that excessive division (i.e., 16) results in bad performance while appropriate division (i.e., 8) improves the performance.

Benefits of MSFC and MSFA. We demonstrate the effectiveness of MSFC and MSFA by comparing them with one-dimensional convolution and SENet [11], as shown in Table 7.

Table 7: Ablation study results on CULane with ResNet-18 backbone. FC and FA denote feature compression and aggregation processing (see Section 3.3 and Section 3.4). Conv means one-dimensional convolution. SE means SENet [11], which uses GAP to quantify feature channels.

FC	FA	F1 (%)	FPS	MACs(G)
Conv	-	75.12	246	9.409
MSFC	-	75.76	248	9.402
Conv	MSFA	75.28	236	9.411
MSFC	MSFA	**76.32**	237	9.404
SE	SE	75.44	238	9.404
SE	MSFA	75.62	238	
MSFC	SE	75.69	237	
MSFC	MSFA	**76.32**	237	

At the first part of Table 7, after replacing the one-dimensional convolution with MSFC for feature compression, a F1 score gain of 0.64 % with less calculation is achieved, which shows the effectiveness of MSFC. MSFA further improves the performance from 75.76 % to 76.32 % in terms of F1 score, with only a small amount of calculation increase. It is worth noting that adding MSFA on the basis of one-dimensional convolution does not improve the performance too much, indicating that the information loss of the one-dimensional convolution hinders the upper bound of the performance.

At the second part of Table 7, we further compare the performance of SENet with our proposed modules (i.e., MSFC and MSFA). Specifically, we compare GAP (the lowest frequency of DCT) with multiple frequency descriptions of DCT. It can be clearly seen that there is a significant improvement after exploiting MSFC and MSFA. The multi-spectral analysis obtained by DCT can indeed increase the diversity of features, thereby retaining more information and improving performance.

5 CONCLUSION

In this paper, we introduce DCT frequency analysis into lane detection to achieve high speed while maintaining accuracy. Specifically, to address the information loss problem during high-dimensional feature compression, we propose MSFC to exploit multi-spectral descriptions to enhance feature diversity. Moreover, to enhance the communication between pixels in each lane, we propose MSFA based on 1D DCT to perform weighted summation over a variety of frequencies. Our MSLD achieves the state-of-the-art performance on three lane detection benchmarks, i.e., TuSimple, CULane, and LLAMAS. Moreover, by adding MSFC to UFLD, an F1 score gain of 1.18 % with faster speed is achieved.

6 ACKNOWLEDGEMENTS

This work was supported in part by the National Key Research and Development Program of China (No.2018YFB0204301), the National Natural Science Foundation of China (No. 61772380), Major Project for Technological Innovation of Hubei Province (No. 2019AAA044), and Science & Technology Major Project of Hubei Province (Next-Generation AI Technologies, No. 2019AEA170).

REFERENCES

[1] Karsten Behrendt and Ryan Soussan. 2019. Unsupervised labeled lane marker dataset generation using maps. In *Proceedings of the IEEE International Conference on Computer Vision*, Vol. 3. 3–3.

[2] Zhenpeng Chen, Qianfei Liu, and Chenfan Lian. 2019. PointLaneNet: Efficient end-to-end CNNs for Accurate Real-Time Lane Detection. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2563–2568.

[3] Prema M Daigavane and Preeti R Bajaj. 2010. Road lane detection with improved canny edges using ant colony optimization. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. IEEE, 76–80.

[4] Max Ehrlich and Larry S Davis. 2019. Deep residual learning in the jpeg transform domain. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3484–3493.

[5] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[6] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. 2018. Faster neural networks straight from jpeg. *Advances in Neural Information Processing Systems* 31 (2018), 3933–3944.

[7] S Han, H Mao, and WJ Dally. 2015. Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint* (2015).

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[9] Yuenan Hou, Zheng Ma, Chunxiao Liu, Tak-Wai Hui, and Chen Change Loy. 2020. Inter-region affinity distillation for road marking segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12486–12495.

[10] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. 2019. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1013–1021.

[11] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. 2019. Line-CNN: End-to-End Traffic line detection with line proposal unit. *IEEE Transactions on Intelligent Transportation Systems* 21, 1 (2019), 248–258.

[14] Zhenguo Li. 2020. CurveLane-NAS: Unifying Lane-Sensitive Architecture Search and Adaptive Point Blending. (2020).

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

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

[17] Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, and Haowei Li. 2020. Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1394–1399.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[19] Hiren M Mandalia and Mandalia Dario D Salvucci. 2005. Using support vector machines for lane-change detection. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 49. SAGE Publications Sage CA: Los Angeles, CA, 1965–1969.

[20] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11264–11272.

[21] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2018. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[22] Jonah Philion. 2019. 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*. 11582–11591.

[23] Zequn Qin, Huanyu Wang, and Xi Li. 2020. Ultra fast structure-aware deep lane detection. *arXiv preprint arXiv:2004.11757* (2020).

[24] Zequn Qin, Pengyi Zhang, Fei Wu, and Xi Li. 2020. FcaNet: Frequency Channel Attention Networks. *arXiv preprint arXiv:2012.11879* (2020).

[25] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[26] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. 2020. PolyLaneNet: Lane estimation via deep polynomial regression. *arXiv preprint arXiv:2004.10924* (2020).

[27] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixão, Claudine Badue, Alberto F De Souza, and Thiago Olivera-Santos. 2020. Keep your Eyes on the Lane: Attention-guided Lane Detection. *arXiv preprint arXiv:2010.12035* (2020).

[28] TuSimple. 2020. Tusimple benchmark. https://github.com/TuSimple/tusimple-benchmark. *Accessed* (2020).

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[30] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8612–8620.

[31] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*. 3–19.

[32] Seungwoo Yoo, Hee Seok Lee, Heesoo Myeong, Sungrack Yun, Hyoungwoo Park, Janghoon Cho, and Duck Hoon Kim. 2020. End-to-end lane marker detection via row-wise classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 1006–1007.

[33] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. 2020. RESA: Recurrent Feature-Shift Aggregator for Lane Detection. *arXiv preprint arXiv:2008.13719* (2020).