

# Learning Lightweight Lane Detection CNNs by Self Attention Distillation

Yuenan Hou<sup>1</sup>, Zheng Ma<sup>2</sup>, Chunxiao Liu<sup>2</sup>, and Chen Change Loy<sup>3Y</sup>

<sup>1</sup>The Chinese University of Hong Kong <sup>2</sup>SenseTime Group Limited <sup>3</sup>Nanyang Technological University  
hy117@ie.cuhk.edu.hk, {mazheng, liuchunxiao}@sensetime.com, ccloy@ntu.edu.sg

## Abstract

*Training deep models for lane detection is challenging due to the very subtle and sparse supervisory signals inherent in lane annotations. Without learning from much richer context, these models often fail in challenging scenarios, e.g., severe occlusion, ambiguous lanes, and poor lighting conditions. In this paper, we present a novel knowledge distillation approach, i.e., Self Attention Distillation (SAD), which allows a model to learn from itself and gains substantial improvement without any additional supervision or labels. Specifically, we observe that attention maps extracted from a model trained to a reasonable level would encode rich contextual information. The valuable contextual information can be used as a form of ‘free’ supervision for further representation learning through performing top-down and layer-wise attention distillation within the network itself. SAD can be easily incorporated in any feed-forward convolutional neural networks (CNN) and does not increase the inference time. We validate SAD on three popular lane detection benchmarks (TuSimple, CULane and BDD100K) using lightweight models such as ENet, ResNet-18 and ResNet-34. The lightest model, ENet-SAD, performs comparatively or even surpasses existing algorithms. Notably, ENet-SAD has  $20 \times$  fewer parameters and runs  $10 \times$  faster compared to the state-of-the-art SCNN [16], while still achieving compelling performance in all benchmarks. Our code is available at <https://github.com/cardwiny/Codes-for-Lane-Detection>.*

## 1. Introduction

Lane detection [1] plays a pivotal role in autonomous driving as lanes could serve as significant cues for constraining the maneuver of vehicles on roads. Detecting lanes in-the-wild is challenging due to poor lighting conditions, occlusions caused by other vehicles, irrelevant road markings, and the inherent long and thin property of lanes.

Contemporary algorithms [5, 8, 14, 16] typically adopt

a dense prediction formulation, i.e., treat lane detection as a semantic segmentation task, where each pixel in an image is assigned with a binary label to indicate whether it belongs to a lane or not. These methods heavily rely on the segmentation maps of lanes as the supervisory signals. Since lanes are long and thin, the number of annotated lane pixels is far fewer than the background pixels. Learning from such subtle and sparse annotations becomes a major challenge in training deep models for the task. A plausible way is to increase the width of lane annotations. However, it may degrade the detection performance.

Several schemes have been proposed to relieve the reliance of deep models on the sparse annotations, e.g., multi-task learning (MTL) and message passing (MP). For example, Lee *et al.* [14] exploit vanishing points to guide the training of deep models and Pan *et al.* [16] incorporate spatial MP in their lane detection models. MTL can indeed provide additional supervisory signals but it requires additional efforts, usually with human intervention, to prepare the annotations, e.g., scene segmentation maps, vanishing points, or drivable areas. MP can help propagate the information between neurons to counter the effect of sparse supervision and better capture the scene context. However, it increases the inference time significantly due to the overhead of MP. For instance, applying MP in a layer of SCNN [16] contributes 35% of its total feed-forward time.

In this work, we present a simple yet novel approach that allows a lane detection network to reinforce representation learning of itself without the need of additional labels and external supervisions. In addition, it does not increase the inference time of the base model. Our approach is named *Self-Attention Distillation* (SAD). As the name implies, SAD allows a network to exploit attention maps derived from its own layers as the distillation targets for its lower layers. Such an attention distillation mechanism is used to complement the usual segmentation-based supervised learning.

SAD is motivated by an interesting observation – when a lane detection network is trained to a reasonable level, attention maps derived from different layers would capture diverse and rich contextual information that hints the lane

Y: Corresponding author.

Figure 1. Attention maps of the ENet [17] before and after applying self attention distillation. Here, we extract the attention maps from the four stages/blocks following the design of ENet model. Note that self attention distillation is added in the 40 K episodes.

locations and a rough outline of the scene, as shown in Fig. 1 (before SAD at 40K episodes). By adding SAD to the learning of this half-trained model, *i.e.*, having the preceding block to mimic the attention maps of a deeper block, *e.g.*, block 3  $\xrightarrow{\text{mimic}}$  block 4 and block 2  $\xrightarrow{\text{mimic}}$  block 3, the network can learn to strengthen its representations, as shown in Fig. 1 (after SAD): (1) the attention maps of lower layers are refined, with richer scene contexts captured by the visual attention, and (2) the better representation learned at lower layers in turn benefits the deeper layers. For instance, although block 4 does not learn from any distillation targets, its representation is reinforced, as evident from the much distinct attention at the lane locations. By contrast, without using SAD, the visual attentions of different layers of the same network hardly improve despite continual training up to 60K episodes.

SAD opens a new possibility of training accurate lane detection networks apart from deploying existing techniques such as multi-task learning and message passing, which can be expensive. It allows us to train small networks with excellent visual attention that is on par with very deep networks. In our experiments, we successfully demonstrate the effectiveness of SAD on a few popular lightweight models, *e.g.*, ENet [17], ResNet-18 [10] and ResNet-34 [10].

In summary, our contributions are three-fold: (1) We propose a novel attention distillation approach, *i.e.*, SAD, to enhance the representation learning of CNN-based lane detection models. SAD is only used in the training phase and brings no computational cost during the deployment. Our work is the first attempt of using a network’s own attention maps as the distillation targets. (2) We carefully and systematically investigate the inner mechanism of SAD, the consideration of choosing among different layer-wise mimicking paths, and the timepoint of introducing SAD to

the training process for improved gains. (3) We verify the usefulness of SAD on boosting the performance of small lane detection networks. We further present several architectural reformulations to ENet [17] for improved performance. Our lightweight model, ENet-SAD, achieves state-of-the-art lane detection performance on TuSimple [18], CULane [16] and BDD100K [23]. It can serve as a strong backbone to facilitate future research on lane detection.

## 2. Related Work

**Lane detection.** Lane detection is conventionally handled via using specialized and hand-crafted features to obtain lane segments. These segments are further grouped to get the final results [2, 6]. These methods have many shortcomings, *e.g.*, requiring complex feature selection process, being lack of robustness and only applicable to relatively easy driving scenarios.

Recently, deep learning has been employed to omit hand-crafted features altogether and learn to extract features in an end-to-end manner [14, 16, 8, 5]. These approaches usually adopt the dense prediction formulation, *i.e.*, treat lane detection as a semantic segmentation task, where each pixel in an image is assigned with a label to indicate whether it belongs to a lane or not. For example, He *et al.* [9] propose Dual-View CNN (DVCNN) to handle lane detection. The method takes front-view and top-view images as inputs. Another popular paradigm performs lane detection from the perspective of instance segmentation. For instance, Neven *et al.* [15] divide lane detection into two stages. Specifically, they first perform binary segmentation that differentiates lane pixels and background pixels. These lane pixels are then classified into different lane instances.

Several schemes have been proposed to complement the lane-based supervision and to capture richer scene context,

e.g., multi-task learning and message passing. For example, Zhang *et al.* [25] establish a framework that accomplishes lane boundary segmentation and road area segmentation simultaneously. Geometric constraints that lane boundaries and lane areas constitute the road are also included to further enhance the final performance. Mohsen *et al.* [8] take lane labels as extra inputs and integrate generative adversarial network (GAN) into the original framework so that the segmentation maps resemble labels more. Pan *et al.* [16] perform sequential message passing between the outputs of top-level layers to better exploit the structural information. While the aforementioned methods do bring additional gains to the performance, multi-task learning requires extra annotations and message passing is not efficient since it propagates information in a sequential way. On the contrary, the proposed SAD is free from the requirement of extra annotations and it does not increase the inference time.

**Knowledge and attention distillation.** Knowledge distillation was originally proposed by [11] to transfer the knowledge from large networks to small networks. Commonly in knowledge distillation, a small student network mimics the intermediate outputs of large teacher networks as well as the labels. In [7, 21] the student and teacher networks share the same capacity and mimicking is performed between pairs of layers with same dimensionality. Hou *et al.* [12] also investigate knowledge distillation performed between heterogeneous networks. Recent studies [24, 19] have expanded knowledge distillation to attention distillation. For instance, Sergey *et al.* [24] introduce two types of attention distillation, i.e., activation-based attention distillation and gradient-based attention distillation. In both kinds of distillation, a student network is trained through learning attention maps derived from a teacher network. The proposed SAD differs to [24] in that our method does not need a teacher network. Distillation is conducted in a layer-wise and top-down manner, in which attention knowledge is propagated layer by layer. This is new in the literature. It is noteworthy that our focus is to investigate the possibility of distilling layer-wise attention for self-learning. This differs from existing studies on using visual attention for weighting features [4, 13, 24].

### 3. Methodology

Lane detection is commonly formulated as a semantic segmentation task. More specifically, given an input image  $\mathbf{X}$ , the objective is to assign a label  $l_{ij}$  ( $l_{ij} = 1, \dots, N_c$ ) to each pixel  $(i, j)$  of  $\mathbf{X}$ , comprising the segmentation map  $\mathbf{s}$ . Here,  $N_c$  is the number of classes. The objective is to learn a mapping  $F: \mathbf{X} \rightarrow \mathbf{s}$ . Recent studies use CNN as  $F$  for end-to-end prediction. The task of lane existence prediction is also introduced to facilitate the evaluation process. We use  $\mathbf{b}$  to represent the binary labels that indicate the existence of lanes. Then, the function becomes  $F: \mathbf{X} \rightarrow (\mathbf{s}, \mathbf{b})$ .

#### 3.1. Self Attention Distillation

Apart from training our lane detection network with the aforementioned semantic segmentation and lane existence prediction losses, we aim to perform layer-wise and top-down attention distillation to enhance the representation learning process. The proposed SAD does not require any external supervision or additional labels since attention maps are derived from the network itself.

In general, attention maps can be divided into two categories, i.e., activation-based attention maps [24] and gradient-based attention maps [24]. The activation-based attention maps are obtained via processing the activation output of a specific layer while the gradient-based ones are obtained via using the layer's gradient output. In the experiment, we empirically find that activation-based attention distillation yields considerable performance gains while gradient-based attention distillation barely works. Hence, in the following sections we only discuss the activation-based attention distillation.

**Activation-based attention distillation.** We use  $A_m \in \mathbb{R}^{C_m \times H_m \times W_m}$  to denote the activation output of the  $m$ -th layer of the network, where  $C_m$ ,  $H_m$  and  $W_m$  denote the channel, height and width, respectively. Let  $M$  denote the number of layers in the network. The generation of the attention map is equivalent to finding a mapping function  $G: \mathbb{R}^{C_m \times H_m \times W_m} \rightarrow \mathbb{R}^{H_m \times W_m}$ . The absolute value of each element in this map represents the importance of this element on the final output. Therefore, this mapping function can be constructed via computing statistics of these values across the channel dimension. More specifically, the following three operations [24] can serve as the mapping function:  $G_{\text{sum}}(A_m) = \sum_{i=1}^{C_m} |A_{mi}|$ ,  $G_{\text{sum}}^p(A_m) = \sum_{i=1}^{C_m} |A_{mi}|^p$  and  $G_{\text{max}}^p(A_m) = \max_{i=1, \dots, C_m} |A_{mi}|^p$ . Here,  $p > 1$  and  $A_{mi}$  denotes the  $i$ -th slice of  $A_m$  in the channel dimension.

The differences between these mapping functions are depicted in Fig. 2. Compared with  $G_{\text{sum}}(A_m)$ ,  $G_{\text{sum}}^p(A_m)$  puts more weights to areas with higher activations. The larger the  $p$  is, the more focus is placed on these highly activated areas. Compared with  $G_{\text{max}}^p(A_m)$ ,  $G_{\text{sum}}^p(A_m)$  is less biased since it calculates weights across multiple neurons instead of selecting the maximum value of these neuron activations as the weight. In the experiment, we empirically find that using  $G_{\text{sum}}^2(\cdot)$  as the mapping function yields the most performance gains.

Figure 2. Attention maps of the block 4 of the ENet model using different mapping functions.

Figure 3. An instantiation of using SAD.  $E_1 \sim E_4$  comprise the encoder of ENet [17],  $D_1$  and  $D_2$  comprise the decoder of ENet. Following [16], we add a small network to predict the existence of lanes, denoted as  $P_1$ . AT-GEN is the attention generator.

**Adding SAD to training.** The intuition behind SAD is that the attention maps of previous layers can distil useful contextual information from those of successive layers. Following [24], we also perform spatial softmax operation  $(\cdot)$  on  $G_{\text{sum}}^2(A_m)$ . Bilinear upsampling  $B(\cdot)$  is added before the softmax operation if the size of original attention maps is different from that of targets. However, different from Sergey *et al.* [24] who perform attention distillation within two networks, the proposed self attention distillation is performed within the network itself.

Adding SAD to an existing network is straight-forward. It is possible to introduce SAD at different timepoint of the training, which could affect the convergence time. We will show an evaluation in the experiment section. Here we assume an ENet half-trained to 40K episodes. As shown in Fig. 3, we add an attention generator, abbreviated as AT-GEN, after each  $E_2$ ,  $E_3$ , and  $E_4$  encoder block of ENet. Formally, AT-GEN is represented by a function  $(\cdot) = (B(G_{\text{sum}}^2(\cdot)))$ . A successive layer-wise distillation loss is formulated as follows:

$$L_{\text{distill}}(A_m, A_{m+1}) = \sum_{m=1}^{M-1} L_d(A_m, A_{m+1}), \quad (1)$$

where  $L_d$  is typically defined as a  $L_2$  loss and  $A_{m+1}$  is the target of the distillation loss. In the example shown in Fig. 3, we have the number of layers  $M = 4$ . Note that we do not assign different weights to different SAD paths, although this is possible. We found that this uniform scheme works well in our experiments.

The total loss is comprised of four terms:

$$L = \underbrace{L_{\text{seg}}(S, \hat{S})}_{\text{segmentation loss}} + \underbrace{L_{\text{IoU}}(S, \hat{S})}_{\text{IoU loss}} + \underbrace{L_{\text{exist}}(\mathbf{b}, \hat{\mathbf{b}})}_{\text{existence loss}} + \underbrace{L_{\text{distill}}(A_m, A_{m+1})}_{\text{distillation loss}}. \quad (2)$$

Here, the first two terms are segmentation losses that comprise of the standard cross entropy loss  $L_{\text{seg}}$  and the IoU

loss  $L_{\text{IoU}}$ . The IoU loss aims at increasing the intersection-over-union between the predicted lane pixels and ground-truth lane pixels. It is formulated as  $L_{\text{IoU}} = 1 - \frac{N_p}{N_p + N_g - N_o}$ , where  $N_p$  is the number of predicted lane pixels,  $N_g$  is the number of ground-truth lane pixels and  $N_o$  is the number of lane pixels in the overlapped areas between predicted lane areas and ground-truth lane areas.  $L_{\text{exist}}$  is the binary cross entropy loss.  $\hat{S}$  is the segmentation map produced by the network and  $\hat{\mathbf{b}}$  is the prediction of the existence of lanes. The parameters  $\alpha$ ,  $\beta$ , and  $\gamma$  balance the influence of segmentation losses, existence loss, and distillation loss on the final task.

It is noteworthy that the SAD paths can be generalized to dense connections beyond the example shown here. For instance, we can add block 1  $\xrightarrow{\text{mimic}}$  block 3, block 1  $\xrightarrow{\text{mimic}}$  block 4, and block 2  $\xrightarrow{\text{mimic}}$  block 4 in addition to the current paths. In general, the number of possible SAD paths for a network with a depth of  $M$  layers is  $\frac{M(M-1)}{2}$ . We will evaluate this possibility in our experiments.

### Visualization of attention maps with and without SAD.

We investigate the influence of SAD by studying the attention maps of different blocks in ENet with and without SAD. More results will be reported in Section 4. Both networks with and without SAD are trained up to 60K episodes. We visualize the attention maps of four existing blocks in ENet. As can be observed in Fig. 4, after adding SAD, the attention maps of ENet become more concentrated on task-relevant objects, *e.g.*, lanes, vehicles and road curbs. This would in turn improve the lane detection accuracy, as we will show in the experiments.

## 3.2. Lane Prediction

The output of the model is not post-processed for TuSimple and BDD100K except CULane. For CULane, in the inference stage, we feed the image into the ENet model. Then the multi-channel probability maps and the lane existence vector are obtained. Following [16], the final output is obtained as follows: First, we use a  $9 \times 9$  kernel to smooth the probability maps. Then, for each lane whose

Figure 4. Attention maps of ENet with and without self attention distillation. Both networks with and without SAD are trained up to 60K episodes. SAD is applied to ENet at 40K training episodes.

existence probability is larger than 0.5, we search the corresponding probability map every 20 rows for the position with the highest probability value. In the end, we use cubic splines to connect these positions to get the final output.

### 3.3. Architecture Design

The original ENet model is an encoder-decoder structure comprised of  $E_1 \sim E_4$ ,  $D_1$  and  $D_2$ . Following [16], we add a small network  $P_1$  to predict the existence of lanes. The encoder module is shared to save memory space. Apart from this modification, we also observed some useful techniques to modify ENet for achieving better performance in the lane detection task. Dilated convolution [22] is added to replace the original convolution layers in the lane existence prediction branch to increase the receptive field of the network without increasing the number of parameters. In the original design, the resolution of feature maps of  $E_4$  is only  $36 \times 100$  for CULane. This leads to severe loss of information. Hence, we use feature concatenation to fuse the output of  $E_4$  with that of  $E_3$  so that the output of the encoder can benefit from information encoded in previous layers.

## 4. Experiments

**Datasets.** Figure 5 shows several video frames of three datasets that we use in our experiments. They are TuSimple [18], CULane [16] and BDD100K [23]. TuSimple and CULane are widely used in the literature. Many algorithms [16, 15, 8] have been tested in TuSimple since it was the largest lane detection dataset before 2018. As to CULane, it contains many challenging driving scenarios like crowded road conditions or roads under poor lighting (see Fig. 5). BDD100K is originally designed for lane

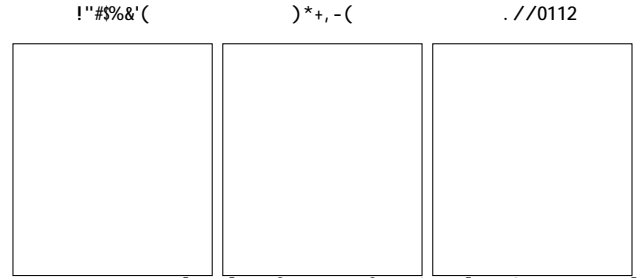


Figure 5. Typical video frames of TuSimple, CULane and BDD100K datasets. Ground-truth lanes are marked in green color.

instance classification. However, since there are typically multiple lanes in an image and these lanes are usually very close to each other, using instance segmentation algorithms will yield inferior performance. Therefore, we choose to only detect lanes without differentiating lane instances for BDD100K. We discuss the details of transforming the original ground truths for our task in the following section on implementation details. Table 1 summarizes their details. Note that the last column of Table 1 shows that TuSimple and CULane have no more than 5 lanes in a video frame while BDD100K typically contains more than 8 lanes in a video frame. Besides, TuSimple is relatively easy while CULane and BDD100K are more challenging considering the total number of video frames and road types. Note that the original BDD100K dataset provides 100K video frames, in which 70K are used for training, 10K for validation and 20K for testing. However, since the ground-truth labels of the testing partition are not publicly available, we keep the training set unchanged but use the original validation set for testing. A new validation set is allocated separately from the training set, as shown in Table 1.



Table 1. Basic information of three lane detection datasets.

Name	# Frame	Train	Validation	Test	Resolution	Road Type	# Lane > 5 ?
TuSimple [18]	6, 408	3, 268	358	2, 782	1280 × 720	highway	×
CULane [16]	133, 235	88, 880	9, 675	34, 680	1640 × 590	urban, rural and highway	×
BDD100K [23]	80, 000	60, 000	10, 000	10, 000	1280 × 720	urban, rural and highway	

**Evaluation metrics.** To facilitate comparisons against previous studies, we follow the literature and use the corresponding evaluation metrics for each particular dataset.

1) *TuSimple*. We use the official metric (accuracy) as the evaluation criterion. Besides, false positive (FP) and false negative (FN) are also reported. Accuracy is computed as [18]:  $\text{Accuracy} = \frac{N_{\text{pred}}}{N_{\text{gt}}}$ , where  $N_{\text{pred}}$  is the number of correctly predicted lane points and  $N_{\text{gt}}$  is the number of ground-truth lane points.

2) *CULane*. Following [16], to judge whether a lane is correctly detected, we treat each lane as a line with 30 pixel width and compute the intersection-over-union (IoU) between labels and predictions. Predictions whose IoUs are larger than 0.5 are considered as true positives (TP). Then, we use  $F_1$  measure as the evaluation metric, which is defined as:  $F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ , where  $\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$  and  $\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$ .

3) *BDD100K*. Since there are typically more than 8 lanes in an image, we decide to use pixel accuracy and IoU of lanes to evaluate the performance of different models.

**Implementation details.** Following [16], we resize the images of TuSimple and CULane to 368×640 and 288×800, respectively. As to BDD100K, we resize the image to 360×640 to save memory usage. The lanes of BDD100K are labelled by two lines. Training the networks using the provided labels is tricky. Therefore, based on these two lines, we calculate the center lines as new targets. We dilate ground-truth lanes of the training set of BDD100K as 8 pixels to provide denser targets while keeping these of testing set unchanged (2 pixels). We use SGD [3] to train our models and the learning rate is set to 0.01. Batch size is set as 12 and the total number of training episodes is set as 1800 for TuSimple and 60K for CULane and BDD100K. The cross entropy loss of background pixels is multiplied by 0.4. Loss coefficients  $\lambda_1$ ,  $\lambda_2$ , and  $\lambda_3$  are set as 0.1. Since we select lane pixel accuracy and IoU as the evaluation criterion for BDD100K dataset, we alter the original segmentation branch to output binary segmentation maps to facilitate the evaluation on BDD100K. The lane existence prediction branch is also removed for the BDD100K evaluation.

We empirically found that several practical techniques, *i.e.*, data augmentation and IoU loss, can considerably enhance the performance of CNN-based lane detection models. As to data augmentation, we use random rotation, random cropping and horizontal flipping to process the input images. In our experiments, we apply the same segmentation losses and augmentation strategy to our method,

Table 2. Performance of different algorithms on TuSimple testing set. Here "R-18-SAD" denotes ResNet-18 + SAD and we use the same abbreviation in the following sections.

Algorithm	Accuracy	FP	FN
ResNet-18 [10]	92.69%	0.0948	0.0822
ResNet-34 [10]	92.84%	0.0918	0.0796
ENet [17]	93.02%	0.0886	0.0734
LaneNet [15]	96.38%	0.0780	0.0244
EL-GAN [8]	96.39%	0.0412	0.0336
SCNN [16]	96.53%	0.0617	0.0180
<b>R-18-SAD (ours)</b>	96.02%	0.0786	0.0451
<b>R-34-SAD (ours)</b>	96.24%	0.0712	0.0344
<b>ENet-SAD (ours)</b>	<b>96.64%</b>	0.0602	0.0205

Table 4. Comparative results on BDD100K test set.

Algorithm	Accuracy	IoU
ResNet-18 [10]	30.66%	11.07
ResNet-34 [10]	30.92%	12.24
ResNet-101 [10]	34.45%	15.02
ENet [17]	34.12%	14.64
SCNN [16]	35.79%	15.84
<b>R-18-SAD (ours)</b>	31.10%	13.29
<b>R-34-SAD (ours)</b>	32.68%	14.56
<b>R-101-SAD (ours)</b>	35.56%	15.96
<b>ENet-SAD (ours)</b>	<b>36.56%</b>	<b>16.02</b>

SCNN, ResNet baselines, and deep supervision methods, to ensure a fair comparison. Since the source codes of LaneNet [15] and EL-GAN [8] are not available, we use their results reported in their papers.

## 4.1. Results

Tables 2-4 summarize the performance of our methods, *i.e.*, ResNet-18-SAD, ResNet-34-SAD, and ENet-SAD against state-of-the-art algorithms on the testing set of TuSimple, CULane and BDD100K datasets. We also report the runtime and parameter count of different algorithm in Table 3 so that we can compare the performance with the complexity of the model taken into account. The runtime is recorded using a single GPU (GeForce GTX TITAN X) and the final value of runtime is obtained after averaging the runtime of 100 samples.

It is observed that ENet-SAD outperforms all baselines in BDD100K while achieving compelling performance in TuSimple and CULane. Considering that ENet-SAD has 20 × fewer parameters and runs 10 × faster compared with SCNN on CULane testing set, the performance strongly suggests the effectiveness of SAD. It is observed that ResNet-18-SAD and ResNet-34-SAD achieve slightly inferior performance to ENet-SAD despite their larger model

Table 3. Performance ( $F_1$ -measure) of different algorithms on CULane testing set. For crossroad, only FP is shown. The second column denotes the proportion of each scenario in the testing set.

Category	Proportion	ENet-SAD	R-18-SAD	R-34-SAD	R-101-SAD	ResNet-101 [10]	SCNN [16]
Normal	27.7%	90.1	89.8	89.9	<b>90.7</b>	90.2	90.6
Crowded	23.4%	<b>68.8</b>	68.1	68.5	<b>70.0</b>	68.2	69.7
Night	20.3%	<b>66.0</b>	64.2	64.6	<b>66.3</b>	65.9	66.1
No line	11.7%	41.6	42.5	42.2	<b>43.5</b>	41.7	43.4
Shadow	2.7%	65.9	67.5	<b>67.7</b>	67.0	64.6	<b>66.9</b>
Arrow	2.6%	<b>84.0</b>	83.9	83.8	<b>84.4</b>	84.0	84.1
Dazzle light	1.4%	<b>60.2</b>	59.8	59.9	59.9	59.8	58.5
Curve	1.2%	65.7	65.5	<b>66.0</b>	65.7	65.5	64.4
Crossroad	9.0%	1998	1995	<b>1960</b>	2052	2183	1990
Total	–	70.8	70.5	70.7	<b>71.8</b>	70.8	71.6
Runtime (ms)	–	<b>13.4</b>	25.3	50.5	171.2	171.2	133.5
Parameter (M)	–	<b>0.98</b>	12.41	22.72	52.53	52.53	20.72

capacity. The is because ResNet-18 and ResNet-34 only use spatial upsampling as the decoder while ENet has a specially designed decoder for the task. It is noteworthy that SAD also helps given a deeper model. Specifically, we apply SAD to ResNet-101, and find that it increases the  $F_1$ -measure from 70.8 to 71.8 in CULane and the accuracy increases from 34.45% to 35.56% in BDD100K.

We show some qualitative results of our algorithm and some baselines in these three benchmarks. As can be seen in Fig. 6, ENet-SAD can detect lanes more precisely than ENet [17] in TuSimple and CULane. As can be seen in Fig. 7, the output probability maps of ENet-SAD are more compact and contain less noise compared with those of vanilla ENet and SCNN in poor lighting conditions. However, since many images in BDD100K contain more than 8 lanes and are collected in challenging scenarios like severe occlusion and poor lighting conditions, the performance of all algorithms is unsatisfactory and needs further improvement. In general, SAD can improve the visual attention as well as the detection performance in challenging conditions like crowded roads and poor light conditions.

We also perform experiments that apply SAD and remove the effect of the P1 branch by blocking the gradient of the P1 branch from the main branch. Results show that ENet-SAD (without supervision from P1 branch) can still achieve 96.61% on TuSimple, 70.8 on CULane and 36.54% on BDD100K, which means the performance gains come mainly from SAD itself.

## 4.2. Ablation Study

We investigate the effects of different factors, e.g., the mimicking path, on the final performance. Besides, we also perform extensive experiments to investigate the timepoint to introduce SAD in the training process.

**Distillation paths of SAD.** We summarize the performance of performing SAD between different blocks of ENet in Table 5. We have a few observations. (1) SAD works well in the middle and high-level layers. (2) Adding SAD in

Figure 6. Performance of different algorithms on (a) TuSimple and (b) CULane testing sets. The number below each image denotes the accuracy. Ground-truth lanes are drawn on the input image.

Figure 7. Performance of different algorithms on BDD100K testing set. We visualize the probability maps to better showcase the effect of adding self attention distillation. The brightness of the pixel indicates the probability of this pixel belonging to lanes. The number below each image denotes the pixel accuracy of lanes. Ground-truth lanes are drawn on the input image.

low level layers will degrade the performance. The reason why SAD does not work in low-level layers is that these layers are originally designated to detect low-level details of the scene. Making them to mimic the attention maps of later layers will inevitably harm their ability of detecting local features since later layers encode more global information. Besides, we also find that mimicking the attention

Table 5. Performance of different variants of ENet-SAD on TuSimple testing set.  $P_{ij}$  denotes that the output of the  $i$ -th block of ENet mimics the output of the  $j$ -th block.

Path	Accuracy	Path	Accuracy	Path	Accuracy
$P_{12}$	91.22%	$P_{23}$	94.72%	$P_{23}, P_{24}$	95.38%
$P_{13}$	91.36%	$P_{24}$	94.63%	$P_{23}, P_{34}$	<b>96.64%</b>
$P_{14}$	91.47%	$P_{34}$	95.29%	$P_{24}, P_{34}$	96.52%

maps of the neighbouring layer successively brings more performance gains compared with mimicking those of non-adjacent layers ( $P_{23} + P_{34}$  outperforms  $P_{24} + P_{34}$ ). We conjecture that attention maps of neighbouring layers are closer from the semantic perspective compared with those of non-neighbouring layers (see Fig. 1).

**Backward distillation.** We also tested another distillation scheme that makes higher layers to mimic lower layers. It decreases the performance of ENet from 93.02% to 91.26% in TuSimple dataset. This is not surprising as low-level attention maps contain more details and are more noisy. Having higher-level layers to mimic lower layers will inevitably interfere the global information captured in higher layers, hampering the crucial clues for the lane detection task.

**SAD v.s. Deep Supervision.** We also compare SAD with deep supervision [20]. Here, deep supervision denotes the algorithm that uses the labels directly as supervision for each layer in the network. More specifically, we use 1x1 convolution and bilinear upsampling to obtain the prediction of intermediate layers and use the cross entropy loss to train the intermediate outputs of the model. We empirically find that adding deep supervision in blocks 2 to 4 obtains the most significant performance gains. As can be seen in Table 6, SAD brings more performance gains than deep supervision in all three benchmarks. We attribute this to the following reasons. Firstly, compared with labels that are considered sparse and rigid, SAD provides softer attention targets that capture more contextual information that indicate the scene structure. Distilling information from attention maps of later layers helps previous layers to grasp the contextual signals. Secondly, a SAD path offers a feedback connection from deeper layers to shallower layers. The connection helps facilitate reciprocal learning between successive layers through attention distillation.

**When to add SAD.** Recall that we assume a half-trained model before we add SAD into the training. Here, we investigate the different timepoints to add SAD. As can be seen in Fig. 8, although different timepoints of introducing SAD achieve almost the same performance in the end, the time to add SAD has an effect on the convergence speed of the networks. We attribute the phenomenon to the quality of the target attention maps produced by later layers. In earlier training stage, deeper layers have not been trained well and therefore the distillation targets produced by these layers are of low quality. Introducing SAD at these earlier

Table 6. Performance of SAD and deep supervision applied to different base models on TuSimple, CULane and BDD100K testing sets.

Algorithm	TuSimple	CULane	BDD100K	
	Accuracy	Total	Accuracy	IoU
ENet	93.02%	68.4	34.12%	14.64
ENet-Deep	94.69%	69.6	35.61%	15.38
ENet-SAD	<b>96.64%</b>	<b>70.8</b>	<b>36.56%</b>	<b>16.02</b>
R-18	92.69%	67.9	30.66%	11.07
R-18-Deep	94.14%	68.8	30.95%	12.23
R-18-SAD	<b>96.02%</b>	<b>70.5</b>	<b>31.10%</b>	<b>13.29</b>
R-34	92.84%	68.6	30.92%	12.24
R-34-Deep	94.52%	69.2	31.72%	13.59
R-34-SAD	<b>96.24%</b>	<b>70.7</b>	<b>32.68%</b>	<b>14.56</b>

Figure 8. Performance of adding self attention distillation on the ENet model at different training episodes on the CULane validation set. The number in the legend denotes the episode when self attention distillation is added. "Baseline" denotes the ENet model without self attention distillation.

stages is not as fruitful. Conversely, adding SAD in later training stage would benefit the representation learning of the previous layers.

## 5. Discussion

We have proposed a simple yet effective attention distillation approach, *i.e.*, SAD, to improve the representation learning of CNN-based lane detection models. SAD is validated in various models (*i.e.*, ENet, ResNet-18, ResNet-34, and ResNet-101) and achieves consistent performance gains in three popular benchmarks (*i.e.*, TuSimple, CULane and BDD100K), demonstrating the effectiveness of SAD. The results show that SAD can generally improve the visual attention of different layers in various networks. It would be interesting to extend this idea to other tasks that demands fine-grained attention to details, such as image saliency detection and image matting.

**Acknowledgement:** This work is supported by SenseTime Group Limited, the General Research Fund sponsored by the Research Grants Council of the Hong Kong SAR (CUHK 14241716), Singapore MOE AcRF Tier 1 (M4012082.020), NTU SUG, and NTU NAP.



## References

- [1] Massimo Bertozzi and Alberto Broggi. Gold: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1):62–81, 1998.
- [2] Amol Borkar, Monson Hayes, and Mark T Smith. A novel lane detection system with efficient ground truth generation. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):365–374, 2012.
- [3] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–186. Springer, 2010.
- [4] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: scale-aware semantic image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3640–3649, 2016.
- [5] Zhe Chen and Zijing Chen. Rbnet: A deep neural network for unified road and road boundary detection. In *International Conference on Neural Information Processing*, pages 677–687. Springer, 2017.
- [6] Hendrik Deusch, Jürgen Wiest, Stephan Reuter, Magdalena Szczot, Marcus Konrad, and Klaus Dietmayer. A random finite set approach to multiple lane detection. In *IEEE Conference on Intelligent Transportation Systems*, pages 270–275. IEEE, 2012.
- [7] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *International Conference on Machine Learning*, pages 1602–1611, 2018.
- [8] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. EL-GAN: embedding loss driven generative adversarial networks for lane detection. In *European Conference on Computer Vision*, pages 256–272. Springer, 2018.
- [9] Bei He, Rui Ai, Yang Yan, and Xianpeng Lang. Accurate and robust lane detection based on dual-view convolutional neural network. In *IEEE Intelligent Vehicles Symposium*, pages 1041–1046. IEEE, 2016.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *STAT*, 1050:9, 2015.
- [12] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning to steer by mimicking features from heterogeneous auxiliary networks. In *Association for the Advancement of Artificial Intelligence*, 2019.
- [13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.
- [14] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon. Vpnet: Vanishing point guided network for lane and road marking detection and recognition. In *IEEE International Conference on Computer Vision*, pages 1965–1973. IEEE, 2017.
- [15] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *IEEE Intelligent Vehicles Symposium*, pages 286–291. IEEE, 2018.
- [16] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial CNN for traffic scene understanding. In *Association for the Advancement of Artificial Intelligence*, 2018.
- [17] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.
- [18] TuSimple. <http://benchmark.tusimple.ai/#/t/1>. Accessed: 2018-09-08.
- [19] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *IEEE Conference on Computer Vision and Pattern*, 2017.
- [20] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *IEEE International Conference on Computer Vision*, 2015.
- [21] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.
- [22] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.
- [23] 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. *arXiv preprint arXiv:1805.04687*, 2018.
- [24] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.
- [25] Jie Zhang, Yi Xu, Bingbing Ni, and Zhenyu Duan. Geometric constrained joint lane segmentation and lane boundary detection. In *European Conference on Computer Vision*, pages 486–502, 2018.