A Hybrid Spatial-temporal Deep Learning Architecture for Lane Detection

Yongqi Dong [1, *], Sandeep Patil [2], Bart van Arem [1], Haneen Farah [1]

[1]Department of Transport and Planning, Faculty of Civil Engineering and Geosciences, Delft University of Technology, 2628 CN Delft, The Netherlands
[2]Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, 2628 CN Delft, The Netherlands

Abstract: Reliable and accurate lane detection is of vital importance for the safe performance of Lane Keeping Assistance and Lane Departure Warning systems. However, under certain challenging peculiar circumstances, it is difficult to get satisfactory performance in accurately detecting the lanes from one single image which is often the case in current literature. Since lane markings are continuous lines, the lanes that are difficult to be accurately detected in the single current image can potentially be better deduced if information from previous frames is incorporated. This study proposes a novel hybrid spatial-temporal sequence-to-one deep learning architecture making full use of the spatial-temporal information in multiple continuous image frames to detect lane markings in the very last current frame. Specifically, the hybrid model integrates the single image feature extraction module with the spatial convolutional neural network (SCNN) embedded for excavating spatial features and relationships in one single image, the spatial-temporal feature integration module with spatial-temporal recurrent neural network (ST-RNN), which can capture the spatial-temporal correlations and time dependencies among image sequences, and the encoder-decoder structure, which makes this image segmentation problem work in an end-to-end supervised learning format. Extensive experiments reveal that the proposed model can effectively handle challenging driving scenes and outperforms available state-of-the-art methods with a large margin.

1. Introduction

The interest in developing automated driving functionalities, and in the end, fully automated vehicles, has been increasing vastly over the last decade. The safety of these automated functionalities is a crucial element that continues to be a priority for academic researchers, manufacturers, policymakers, and their potential future users. The ultimate goal is to arrive at a full understanding of the environment around the automated vehicle through the use of various sensors and control modules. Visual-based methods have lately been boosted by advancements in computer vision and machine learning. Regarding environmental perception, camera-based lane detection is important as it allows the vehicle to position

Manuscript received October 1, 2021.

* Corresponding author. Email: y.dong-4@tudelft.nl; Tel: +31 (0) 616619399
Mailing address: Room 4.41, Building 23, Stevinweg 1, 2628 CN Delft, The Netherlands

itself within the lane. This is also the foundation on which most Lane Keeping Assistance and Lane Departure Warning systems are based (Andrade et al., 2019; Bar Hillel et al., 2014; Chen et al., 2020; Liang et al., 2020; Xing et al., 2018).

Traditional vision-based lane-detection methods are based upon hand-crafted low-level features (e.g., color, gradient, and ridge features) and usually work in a way of 4-step procedures, i.e., image pre-processing, feature extraction, line detection and fitting, and post-processing (Bar Hillel et al., 2014; Haris and Glowacz, 2021). Traditional computer vision techniques, e.g., Inverse Perspective Mapping (Aly, 2008; Wang et al., 2014), Hough transform (Berriel et al., 2017; Jiao et al., 2019; Zheng et al., 2018), Gaussian filters (Aly, 2008; Sivaraman and Trivedi, 2013; Wang et al., 2012), and Random Sample Consensus (RANSAC) (Aly, 2008; Choi et al., 2018; Du et al., 2018; Guo et al., 2015; Lu et al., 2019), are usually adopted in the 4-step procedures. The problems of traditional methods are: (a) hand-crafted features take a long time and are not always useful, suitable, or powerful; (b) the detection results are always based upon one single image. Thus the detection accuracies are relatively not high.

During the very last decade, with the advancements in deep learning algorithms and computational power of the hardware, a number of neural network based lane-detection methods have been proposed with good performance. There are generally two dominant approaches (Tabelini et al., 2020b), i.e., (1) segmentation-based pipeline (Kim and Park, 2017; Ko et al., 2020; T. Liu et al., 2020; Pan et al., 2018; Zhang et al., 2021; Zou et al., 2020), in which predictions are made on the per-pixel basis, classifying each pixel as either lane or not; (2) the pipeline using row-based prediction (Hou et al., 2020; Qin et al., 2020; Yoo et al., 2020), in which the image is split into (horizontal) grid divisions and the model predicts the most probable location to contain a part of a lane marking for each row. Recently, Liu et al. (2021) summarized two more categories of deep learning based lane detection methods, i.e., the anchor-based approach (Chen et al., 2019; Li et al., 2020; Tabelini et al., 2020b; Xu et al., 2020), which focuses on optimizing the line shape by regressing the relative coordinates with the help of predefined anchors, and the parametric prediction based method that directly outputs parametric lines expressed by curve equation (R. Liu et al., 2020; Tabelini et al., 2020a). Apart from the dominant approaches, some other less common methods were also proposed recently. For instance, Philion (2019) developed a novel learning-based approach with a fully convolutional model to decode the lane structures directly instead of delegating structure inference to post-processing, plus an effective approach to adapt the model to new environments using unsupervised style transfer.

Similar to traditional vision-based lane-detection methods, most of the available deep learning models utilize only the current image frame to do the detection. Until very recently, a few studies have explored the combination of CNNs and RNNs to detect lane markings through continuous driving scenes (Zhang et al., 2021; Zou et al., 2020). However, the available methods do not take full advantage of the essential properties of the lane as they are usually long continuous line structures, appearing either in solid or dashed lines. Also, they do not yet make full use of the spatial-temporal information together with the correlation and dependencies in the continuous driving frames. Thus, for certain extremely challenging driving scenes, their detection results are still unsatisfactory. For a first glance regarding some of the mentioned methods compared with the proposed model, Fig. 1 shows the lane detection results with

different methods under three selected challenging scenes.

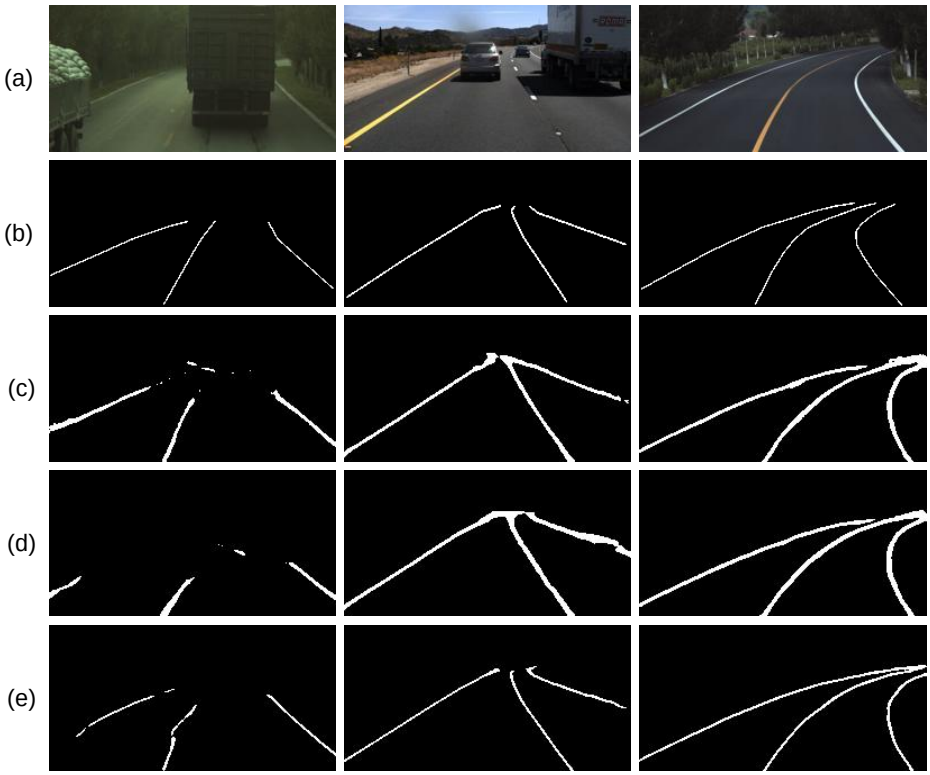


Fig. 1. Lane detection under challenging situations. (a) Three examples of challenging driving scenes. (b) Ground truth. (c) Lane detection results of the SegNet model using a single image. (d) Lane detection results of the SegNet_ConvLSTM model (Zou et al., 2020) using continuous image frames. (e) Lane detection results of the proposed model SCNN_SegNet_ConvLSTM2 using continuous image frames.

In this paper, regarding it as a segmentation task, a novel hybrid spatial-temporal sequence-to-one deep learning architecture is developed for lane detection through a continuous sequence of images and in an end-to-end approach. To cope with challenging driving situations, the hybrid model takes multiple frames of a continuous sequence of images as inputs, and integrates the single image feature extraction module, the spatial-temporal feature integration module, together with the encoder-decoder structure to make full use of the spatial-temporal information in the image sequence. The single image feature extraction module utilizes modified common backbone networks with embedded spatial convolutional neural network (SCNN) (Pan et al., 2018) layers to extract the features in every single image throughout the continuous driving scene. SCNN is powerful in extracting spatial features and relationships in one single image, especially for long continuous shape structures. Then the extracted features are fed to spatial-temporal recurrent neural network (ST-RNN) layers to capture the spatial-temporal correlations and time dependencies among the frames. Encoder-decoder structure is adopted with the encoder consists of SCNN and several fully-convolution layers to downsample the input image and abstract the features, while the decoder, constructed by CNNs, upsample the abstracted outputs of previous layers to the same size as the input image. With the labeled ground truth of the very last image in the continuous frames, the model works in an end-to-end way as supervised learning. To train and validate the proposed

model on two large-scale open-sourced datasets, i.e., tvtLANE (Zou et al., 2020) and TuSimple[1], a corresponding training strategy is also developed. To summarize, the main contributions of this paper lie in:

- A hybrid spatial-temporal sequence-to-one deep neural network architecture integrating the advantages of the encoder-decoder structure, SCNN embedded single image feature extraction module, and ST-RNN module, is proposed;
- The proposed model is the first attempt that tries to strengthen both spatial relation feature extraction in one single image frame and spatial-temporal correlation together with dependencies among continuous image frames for lane detection;
- With the implementations using two commonly used neural network backbones, extensive evaluation experiments on large scale datasets demonstrate the effectiveness and strength of the proposed model architecture;
- The proposed model, making full use of the spatial-temporal information in the continuous image frames, can tackle lane detection in challenging scenes such as curves, dirty roads, serious vehicle occlusions, etc., and outperforms all the available state-of-the-art baseline models with a large margin in both normal and challenging driving scenes.
- Under the proposed architecture, the light version model variant can achieve beyond state-of-the-art performance while using only quite fewer parameters.

[1] TuSimple. Tusimple benchmark. https://github.com/TuSimple/tusimple-benchmark

2. Proposed method

2.1 Overview of the proposed model

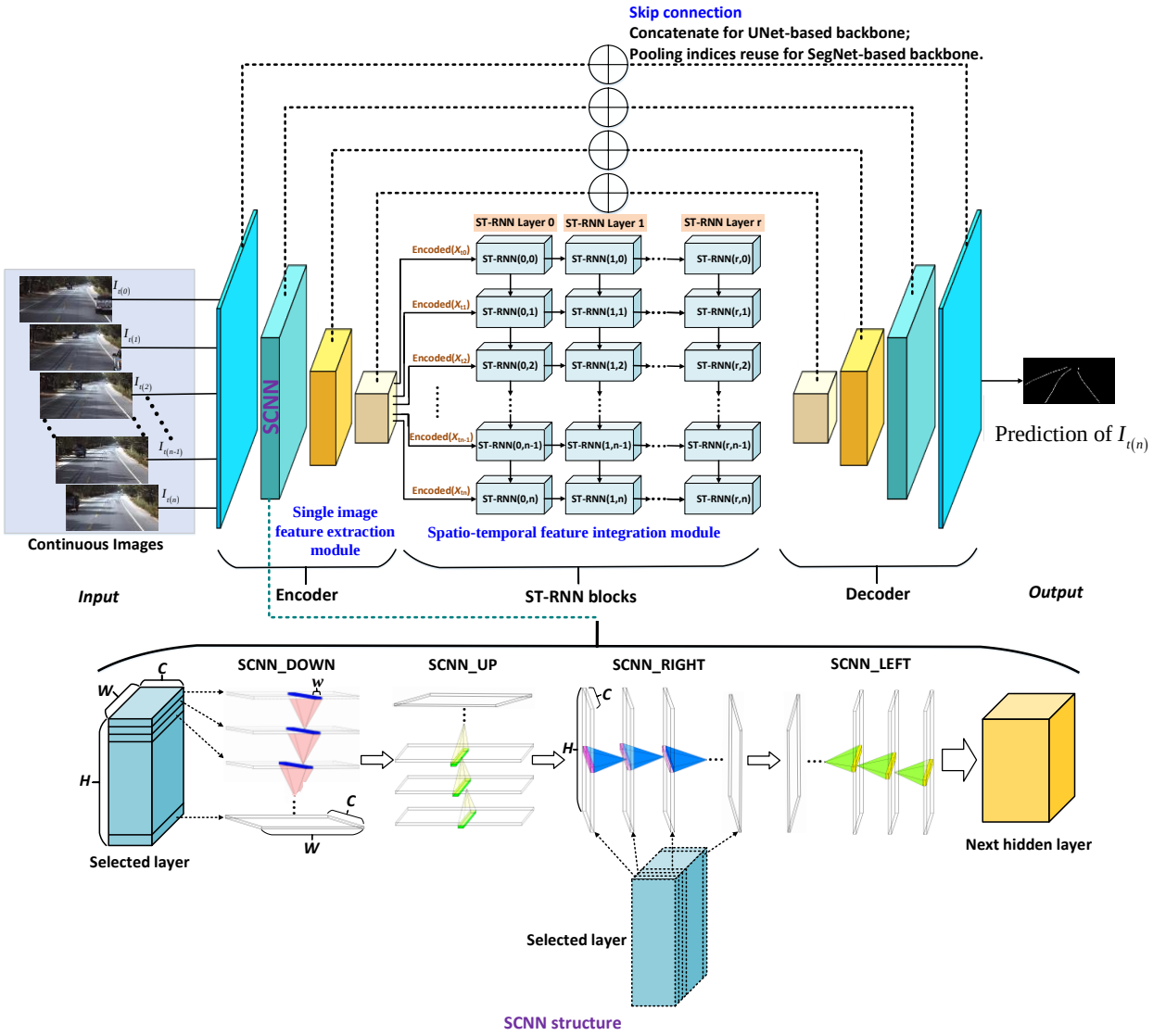


Fig. 2. The architecture of the proposed model.

Although many sophisticated methods have been proposed for lane detection, most of the available methods, using only one single image to do the detection, often cannot deliver satisfactory performance under some extremely challenging scenarios, e.g., dazzle lighting, marking degradation, and serious occlusion. Inspired by the successful precedents of hybrid deep neural network architectures that fuse CNN and RNN to make use of information among continuous multiple frames (Zhang et al., 2021; Zou et al., 2020), plus the domain prior knowledge that traffic lanes are long continuous shape line structure with strong spatial relationship, this study proposes a novel hybrid spatial-temporal sequence-to-one deep neural network architecture by integrating two modules utilizing two distinctive neural networks with complementary merits, i.e., SCNN and ConvLSTM, under an end-to-end encoder-decoder structure, to tackle lane detection in challenging driving scenes.

On the overall level, the proposed model architecture adopts a sequence-to-one end-to-end encoder-decoder structure. The architecture overview is shown in Fig. 2.

Here "sequence-to-one" means the model gets multi images sequences as inputs and output the detection result of the last one image (please note that essentially the model is still utilizing sequence-to-sequence neural networks); "end-to-end" means that the learning algorithm goes directly from the input to the desired output, which refers to the lane detection result in this paper, bypassing the intermediate states (Levinson et al., 2011; Neven et al., 2017); the encoder-decoder structure is a modular structure that consists of an encoder network and a decoder network, and is often employed in sequence-to-sequence tasks, such as language translation e.g., (Sutskever et al., 2014), and speech recognition e.g., (Wu et al., 2017). Here, the proposed model adopts encoder CNN with SCNN layers and decoder CNN using fully convolutional layers. The encoder takes a sequence of continuous image frames, i.e., time-series-images, as input and abstracts the feature map(s) in smaller sizes. To make use of the prior knowledge that traffic lanes are solid- or dashed- line structures with a continuous shape, one special kind of CNN, i.e., SCNN, is adopted after the first CNN hidden layer. With the help of SCNN, spatial features and relationships in every single image will be better extracted. Then the extracted feature maps of the continuous frames, which are constructed in a time-series manner, will be fed to ST-RNN blocks for sequential feature extraction and spatial-temporal information integration. Finally, the decoder network upsamples the abstracted feature maps and decodes the content to the original input image size with the detection results. Note, the proposed model architecture is implemented with two backbones, e.g., UNet (Ronneberger et al., 2015) and SegNet (Badrinarayanan et al., 2017). In the UNet based architecture, similar to (Ronneberger et al., 2015), the proposed model employed the skip connection between the encoder and decoder phase by concatenating operation to reuse features and retained information from previous encoder layers for more accurate predictions; while in the SegNet based networks, at the decoder stage, similar to (Badrinarayanan et al., 2017), the proposed model reused the pooling indices to capture, store, and make use of the vital boundary information in the encoder feature maps. The detailed network implementation is elaborated in the remaining parts in *Section 2*.

2.2 Network design

1) End-to-end encoder-decoder: Regarding lane detection as an image segmentation problem, the encoder-decoder structure based neural network can be implemented and trained in an end-to-end way. Inspired by the excellent performance of Convolutional (CNN)-based Encoder-Decoder for image semantic-segmentation tasks in various domains (Badrinarayanan et al., 2017; Wang et al., 2020; Yasrab et al., 2017), this study also adopted the "symmetrical" CNN-based Encoder-Decoder as the main backbone structure. Convolution and pooling operations are employed to extract and abstract the features in every image in the encoder stage; while in the decoder subset, the inverted convolution and upsampling operation are adopted to grasp the extracted high-order features and construct the outputs layer by layer with regards to the targets. By setting the output target size the same as the input image size, the whole network can work in an end-to-end approach. In the implementation, two widely used architectures, e.g., U-Net and Seg-Net as the network backbone, are adopted. To better extract and make

use of the spatial relations in every image frame, the SCNN layer is introduced in the encoder part of the single image feature extraction module. Furthermore, to excavate and make use of the spatial-temporal correlations and time dependence among the input continuous image frames, ST-RNN blocks are embedded in the middle of the encoder-decoder networks.

2) SCNN: The Spatial Convolutional Neural Network (SCNN) was first proposed by Pan et al. (2018). The "spatial" here means that the specially designed CNN can propagate spatial information via slice-by-slice message passing. The detailed structure of SCNN is demonstrated in the bottom part of Fig. 2.

SCNN can propagate the spatial information in one image through four directions, as shown with the suffix "DOWN", "UP", "RIGHT", "LEFT" in Fig. 2, which denotes SCNN that is downward, upward, rightward, and leftward, respectively. Take the "SCNN_DOWN" module for an example, considering that SCNN is applied on a 3-D tensor of size $C \times H \times W$, where in the lane detection task, C, H, and W denote the number of channels which is 3 for RGB color image, image heights, and image width respectively. For SCNN_D, the input tensor would be split into H slices, and the first slice is then sent into a convolution operation layer with C kernels of size $C \times w$, where w is the kernel width. Different from the traditional CNN in which the output of a convolution layer is fed into the next layer directly, here in SCNN_D the output is added to the next adjacent slice to produce a new slice, and iteratively to the next convolution layer continuing until the last slice is updated. The convolution kernel weights are shared across all slices, and the same mechanism works for other directions of SCNNs.

With the above properties, SCNN has demonstrated its strengths in extracting spatial relationships in the image, which makes it suitable for detecting long continuous shape structures, e.g., traffic lanes, poles, and walls (Pan et al., 2018). However, using only one image to do the detection, SCNN still could not produce satisfying performance under extremely challenging conditions. And that is why a sequence-to-one architecture with continuous image frames as inputs and ST-RNN blocks to capture the spatial-temporal correlations in the continuous frames is proposed in this paper.

3) ST-RNN module: In this proposed framework, we model multiple continuous frames of images as "image-time-series" inputs. To capture the spatial-temporal dependencies and correlations among the image-time-series, the ST-RNN module is embedded in the middle of the encoder-decoder structure, which takes over the output extracted features of the encoder as its input and outputs the integrated spatial-temporal information to the decoder.

Various versions of RNNs have been proposed, e.g., LSTM together with its multivariate version, i.e., fully connected LSTM (FC-LSTM), and Gated Recurrent Unit (GRU), to tackle time-series data in different application domains. In this paper, we employed two state-of-the-art RNN networks, i.e., ConvLSTM (Shi et al., 2015) and Convolutional Gated Recurrent Unit (ConvGRU) (Ballas et al., 2016), which generally outperform other traditional RNN models.

A general critical problem for the vanilla RNN model is the gradients vanishing (Hochreiter and Schmidhuber, 1997; Pascanu et al., 2013; Ribeiro, 2020). For this, LSTM introduces memory cells and gates to control the information flow to trap the gradient preventing it from vanishing during the back-propagation. In LSTM, the information of the new time-series inputs will be accumulated to the memory

cell $\mathcal{C}_t$ if the input gate i_t is on. In contrast, if the information is not "important", the past cell status $\mathcal{C}_{t-1}$ could be "forgotten" by activating the forget gate f_t. Also, there is the output gate o_t which decides whether the latest cell output $\mathcal{C}_t$will be propagated to the final state $\mathcal{H}_t$. The traditional FC-LSTM contains too much redundancy for spatial information, which makes it time-consuming and computational-expensive. To address this, we adopted the ConvLSTM (Shi et al., 2015) as the RNN block in our proposed neural network. In ConvLSTM, the convolutional structures and operations are introduced in both the input-to-state and state-to-state transitions to do spatial information encoding, which also alleviates the problem of time- and computation-consuming.

The key formulation of the ConvLSTM is shown by equation (1)-(5), where $\odot$ denotes the Hadamard product, $*$ denotes the convolution operation, $\sigma(\cdot)$ represents the sigmoid operation, and $\tanh(\cdot)$ represents the hyperbolic tangent non-linearities; X_t, $\mathcal{C}_t$, and $\mathcal{H}_t$ are the input (i.e., the extracted features from the encoder in our framework), memory cell status, and output at time t; i_t, f_t, and o_t are the function values of the input gate, forget gate, and output gate, respectively; W denotes the weight matrices, whose subscripts indicate the two corresponding variables are connected by this matrix. For instance, W_{xc} is the weight matrix between the input extracted features X_t and the memory cell $\mathcal{C}_t$; $'b'$s are biases of the gates, e.g., b_i is the bias of the input gate.

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \odot \mathcal{C}_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \odot \mathcal{C}_{t-1} + b_f) \tag{2}$$

$$\mathcal{C}_t = f_t \odot \mathcal{C}_{t-1} + i_t \odot \tanh(W_{xc} * X_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \tag{3}$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot \mathcal{C}_t + b_o) \tag{4}$$

$$\mathcal{H}_t = o_t \odot \tanh(C_t) \tag{5}$$

The ConvGRU (Ballas et al., 2016) further lightens the computational complexity by reducing a gate structure but could perform similarly or slightly better compared with the traditional RNNs or even ConvLSTM. The procedure of computing different gates and hidden states/outputs of ConvGRU is demonstrated with equation (6)-(9), in which the symbols have the same meaning as described before, while additional z_t and r_t mean the update gate and the reset gate, respectively, plus $\widetilde{\mathcal{H}}$represents the current candidate hidden representation.

$$z_t = \sigma(W_{zx} * X_t + W_{zh} * \mathcal{H}_{t-1} + b_z) \tag{6}$$

$$r_t = \sigma(W_{rx} * X_t + W_{rh} * \mathcal{H}_{t-1} + b_r) \tag{7}$$

$$\widetilde{\mathcal{H}}_t = \tanh(W_{ox} * X_t + W_{oh} * (r_t \odot \mathcal{H}_{t-1}) + b_o) \tag{8}$$

$$\mathcal{H}_t = z_t\widetilde{\mathcal{H}} + (1\text{-}z_t)\mathcal{H}_{t-1} \tag{9}$$

In ConvGRU, there are only two gate structures, i.e., the update gate z_t and the reset gate r_t. It is the update gate z_t that decides to what extent to update the hidden representation when producing the final result of $\mathcal{H}_t$ at the current layer, as shown in equation (9). While the reset gate r_t is served to control how much feature information captured in the previous hidden state should be forgotten by an element-wise multiplication operation when calculating current candidate hidden representation. From the

equations, we can conclude the information of $\mathcal{H}_t$ mainly comes from $\widetilde{\mathcal{H}}_t$, while $\mathcal{H}_{t-1}$as the previous hidden-state representation also participates in the computing process of building the final representation of $\mathcal{H}_t$, thus the temporal dependencies are captured.

In practice, we employed both ConvLSTM and ConvGRU with different numbers of hidden layers to serve as the ST-RNN module in the proposed architecture and tested the corresponding performances, respectively. To be specific, in the proposed network, the input and output sizes of the ST-RNN block are equal to the size of the feature map extracted through the encoder, which are 8 × 16 for UNet based backbones and 4 × 8 for SegNet based backbones. The size of the convolutional kernel in ConvLSTM and ConvGRU is 3 × 3, and each hidden layer has a dimension of 512. The detailed implementations are described in the following section.

2.3 Implementation details

1) Network Details: The proposed spatial-temporal sequence-to-one neural network is developed for the lane detection task with K (K = 5) continuous image frames as inputs. The image frames are firstly input into the convolution blocks of the encoder network for feature extraction and abstraction. Different from the normal CNN-based encoder, we utilized the SCNN layer to effectively extract the spatial relationships within every image. We tested different locations of the SCNN layer, i.e., embedded the SCNN layer after the first hidden convolutional layer or at the very beginning. The outputs of the encoder network are modeled in a time-series manner and fed into the ST-RNN blocks (i.e., ConvLSTM or ConvGRU layers) to further extract more useful and accurate features, especially the temporal dependencies and spatial-temporal correlations among different image frames. In short, the encoder network is primarily responsible for spatial feature extraction and abstraction converting larger images into the specified feature maps, while the ST-RNN blocks accept the extracted features from the continuous image frames in a time-series manner to capture the spatial-temporal dependencies.

The outputs of the RNN blocks are then transferred into the decoder network that adopts deconvolution and upsampling operations to highlight and make full use of the features and rebuild the target to the original size of the input image. Note there is the skip *concatenate* connection (for UNet-based architecture) or pooling indices reusing (for SegNet-based architecture) between the encoder and decoder phase to reuse the retained features from previous encoder layers for more accurate predictions at the decoder phase. After the decoder phase, we obtain the lane detection result as an image in the same size as the original input image frame. With the labeled ground truth and the help of the encoder-decoder structure, the proposed model can be trained and implemented in an end-to-end way. The detailed input, output sizes, together with parameters of the layers in the entire neural network are listed in Appendix Table A1 and Table A2.

For both SegNet-based and UNet-based architecture, we tested two types of RNN layers, i.e., ConvLSTM and ConvGRU, to serve as the RNN block. We also tested the RNN blocks with 1 hidden layer and 2 hidden layers, respectively. So there are four variants of in the proposed SegNet-based models, i.e., SCNN_SegNet_ConvGRU1, SCNN_SegNet_ConvGRU2, SCNN_SegNet_ConvLSTM1, and SCNN_SegNet_ConvLSTM2. SCNN_SegNet_ConvGRU1 means the model is using SegNet as the

backbone with encoder embedding SCNN layers, and 1 hidden layer of ConvGRU as the RNN block. This naming rule applies to the other 3 variants. Also, there are four variants of in the proposed UNet-based models, i.e., SCNN_UNet_ConvGRU1, SCNN_UNet_ConvGRU2, SCNN_UNet_ConvLSTM1, and SCNN_UNet_ConvLSTM2. Similarly, SCNN_UNet_ConvGRU1 means the model is using UNet as the backbone with encoder embedding SCNN layers, and 1 hidden layer of ConvGRU as the RNN block. This naming rule applies to the other 3 model variants.

In the proposed models with U-Net as the backbone, there are some differences regarding the number of kernels used in the last convolutional block in the encoder part compared with the original U-Net's settings. Here, the number of output kernels (channels) of the last convolutional block in the proposed encoder does not double its input kernels, which applies to all the previous convolutional blocks. The main reason is that, similar to (Zou et al., 2020), to better connect the output of the encoder with the RNN block (ConvLSTM or ConvGRU layers), the parameters of the full-connection layer will be quadrupled as the side length of the feature map reduced to half, while the number of kernels remains unchanged. This strategy also somewhat contributes to reducing the parameter size of the whole network.

We also tested a modified light version of UNet (UNetLight) serving as the network backbone to reduce the total parameter size, increase the model's ability to operate in real-time, and also further verify the proposed network architecture's effectiveness. The UNetLight has a similar network design to the demonstration in Table A2. The only difference is that all the numbers of kernels in the ConvBlocks are reduced to half except for the *Input* in *In_ConvBlock* (with the input channel of 3 unchanged) and *Output* in *Out_ConvBlock* (with the output channel of 2 unchanged). To save space, the parameter settings of UNetLight will not be illustrated.

2) Loss function: As the segmentation task and the pixel-wise binary classification problem, cross-entropy is a suitable candidate to serve as the loss function. However, since the pixels classified to be lanes are always quite less than that classified to be the background, meaning that it is an imbalanced binary classification and discriminative segmentation task, in the implementation, we constructed the loss based on the weighted cross-entropy. The adopted loss function as the standard weighted binary cross-entropy function could be given by equation (10).

$$Loss = -\frac{1}{S}\sum_{i=1}^{S}\left[w * y_i * log\left(h_\theta(x_i)\right) + (1\text{-}y_i) * log\left(1 - h_\theta(x_i)\right)\right] \qquad (10)$$

where S is number of training examples, w stands for the weight which is set according to the ratio between the number of pixels in the two classes (i.e., lane and not lane) throughout the whole training set, y_i is the true target label for training example i, x_i is the input for training example i, and h_θ stands for the model with neural network weights θ.

3) Training details: After constructing the proposed neural networks with different variants, we trained them together with the baseline modes on the Dutch high-performance supercomputer clusters, Cartesius and Lisa, using 4 Titan RTX GPUs with the data parallel mechanism in PyTorch, by employing *nn.DataParallel()*. We set the input image size as 128 × 256 to reduce the computational payload. The batch size is set to as large as possible (e.g., 64 for UNet-based network architecture, 100

for SegNet based, and 136 for UNetLight based ones), and the learning rate is initially set to 0.03. The RAdam optimizer (Liu et al., 2019) is first used in our work for training the model at the beginning. And at the later stage, when the training accuracy is beyond 95%, we switch to the Stochastic Gradient Descent (SGD) (Bottou, 2010) optimizer with decay. With the labeled ground truth, the models are trained by iteratively updating the weight parameters and loss based on the deviation between the ground truth and outputs of the proposed network using the backpropagation mechanism.

3. Experiments and results

In this section, extensive experiments were carried out to verify the accuracy, effectiveness, and robustness of our proposed lane detection model using two large-scale open-sourced datasets. The proposed model(s) were evaluated on different driving scenes from different datasets and were qualitatively and quantitatively compared with several state-of-the-art baseline deep learning based lane detection models, e.g., U-Net (Ronneberger et al., 2015), Seg-Net (Badrinarayanan et al., 2017), UNet_1×1 (Zou et al., 2020), SegNet_1×1 (Zou et al., 2020), SCNN (Pan et al., 2018), LaneNet (Neven et al., 2018), UNet_ConvLSTM (Zou et al., 2020), and SegNet_ConvLSTM (Zou et al., 2020).

3.1 Datasets

Table 1.
Trainset and testset in tvtLane

Trainset				
Subset				**Labled Images Num**
Original TuSimple Dataset (Highway)				7,252
Zou et.al. (Zou et al., 2020) Added (Rural Road)				2,296
Sample Methods				
Labled Ground Truth			**Sample Stride**	**Sample Frames**
13^{th}			1	9^{th}, 10^{th}, 11^{th}, 12^{th}, 13^{th}
			2	5^{th}, 7^{th}, 9^{th}, 11^{th}, 13^{th}
			3	1^{st}, 4^{th}, 7^{th}, 10^{th}, 13^{th}
20^{th}			1	16^{th}, 17^{th}, 18^{th}, 19^{th}, 20^{th}
			2	12^{th}, 14^{th}, 16^{th}, 18^{th}, 20^{th}
			3	8^{th}, 11^{th}, 14^{th}, 17^{th}, 20^{th}
Testset				
Subset	**Labled Images Num**	**Labled Ground Truth**	**Sample Stride**	**Test Sample Frames**
Testset #1 Normal	540	13^{th}	1	9^{th}, 10^{th}, 11^{th}, 12^{th}, 13^{th}
		20^{th}	1	16^{th}, 17^{th}, 18^{th}, 19^{th}, 20^{th}
Testset #2 Challenging	728	All	1	1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th} 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 6^{th} 3^{rd}, 4^{th}, 5^{th}, 6^{th}, 7^{th} ...

1) tvtLane training set: To verify the proposed model performance, we first utilized the tvtLane dataset (Zou et al., 2020), which is based upon the Tusimple lane marking challenge dataset, for training, validating, and testing. The original dataset of the Tusimple lane marking challenge consists of 3,626 training and 2,782 testing clips that are collected under different weather conditions and during different

periods. There are 20 continuous frames in each one-second clip saved in the same folder. In each clip, only the lane lines of the 20th frame are labeled with the ground truth officially. To augment the dataset, Zou et.al. (Zou et al., 2020) additionally labeled every 13th image in each clip and added their own collected lane dataset which includes 1,148 sequences of rural road images collected in China. This greatly expands the diversity of the road and driving conditions since the original Tusimple dataset only covers the highway driving conditions. *K* continuous frames (in this paper *K*=5 if not specified) of each clip are used as the inputs with the ground truth of the labeled 13th or 20th frame to train the models.

To further augment the training dataset, rotation, flip, and crop operations were employed, and thus a total number of (3,626 + 1,148) × 4 = 19,096 continuous sequences were produced, in which 38,192 images are labeled with ground truth. To adapt to different driving speeds, the input image sequences are sampled at three different strides with an interval of 1, 2, and 3 frames respectively. Then, there are three sampling ways to construct the training samples with regards to the labeled 13th and 20th frames in each sequence, as demonstrated in Table 1.

2) tvtLane testing set: Regarding the testing set, there are two different datasets, i.e., Testset #1 (normal) and Testset #2 (challenging), which are also formatted with 5 continuous images as the input to detect the lanes in the very last frame with the ground truth. To be specific, Testset #1 is built upon the original TuSimple test set for normal case testing; while Testset #2 is constructed with 12 challenging driving situations, which is especially used for robustness evaluation. The detailed descriptions of the trainset and testset in tvtLane dataset are illustrated in Table 1, with examples shown in Fig. 3.

Fig. 3. Samples data in trainset and testset. (a) original TuSimple dataset (Highway), (b) Zou et al., (2020) added Rural Road situations, (c) Testset #1 Normal situations, and (d) Testset #2 Challenging situations. In each row, the first five images are the input image sequence the last image is the labeled ground truth.

3.2 Qualitative evaluation

Qualitative evaluation with the visualization of the lane detection results is the most intuitive approach for evaluating and comparing the performance of different models, and it helps to find the pros, cons together with insights of the different models.

1) tvtLane Testset #1: normal situations

We present samples of the lane detection results on tvtLane testset #1 of our proposed models and other state-of-the-art deep learning models in Fig. 4. All of the results from the different models are without post-processing.

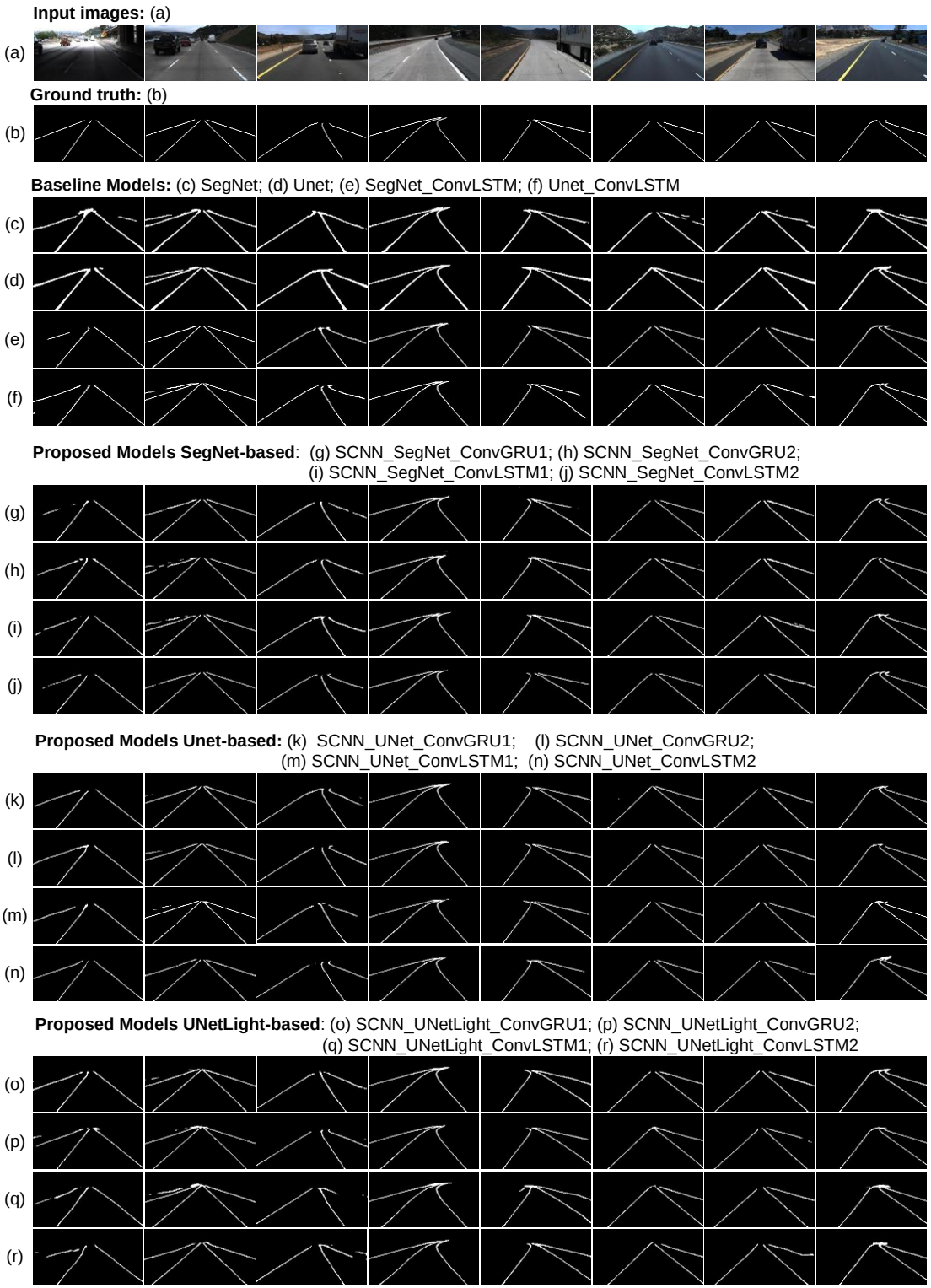


Fig. 4. Qualitative evaluation 1: visual comparison of the lane-detection results on tvtLane Testset #1(normal situations).

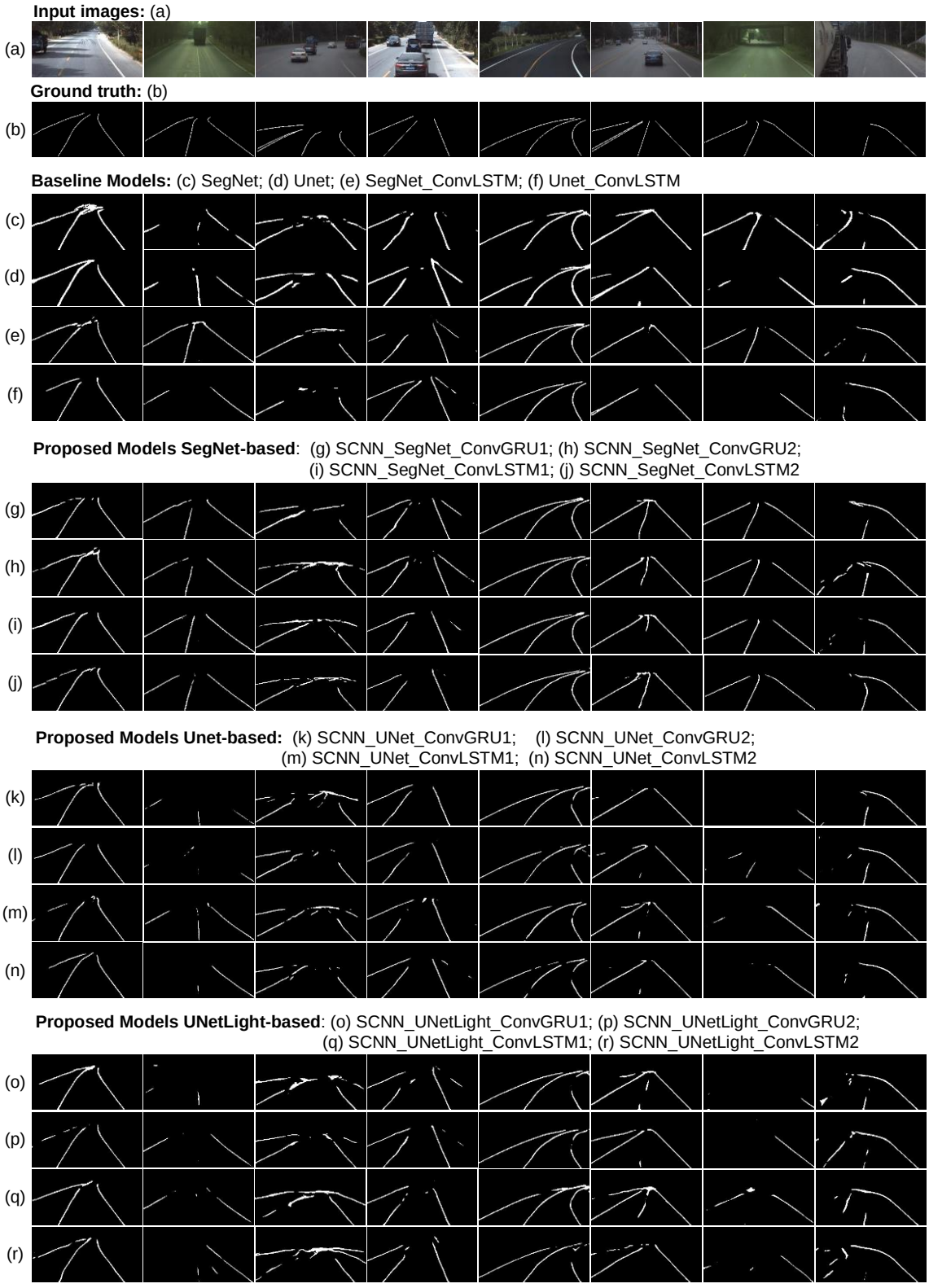


Fig. 5. Qualitative evaluation 2: visual comparison of the lane-detection results on tvtLane Testset #2 (challenging situations).

In general, a good lane detection should include the following 5 properties:

• The number of lane lines should be predicted correctly. A wrong detection or a misprediction may lead the automated vehicles' considering unsafe or unreachable areas as drivable areas, which might

result in accidents. As shown in the 1st and 2nd columns in Fig. 4, the proposed models can detect the correct number of lane lines, while other models, especially the ones using a single image, more or less cannot detect the correct number of lane lines compared with the ground truth.

• The location of each lane line should be predicted precisely the same as the corresponding ground truth. As illustrated in Fig. 4, the proposed models, especially the line (j) with the model named by SCNN_SegNet_ConvLSTM2 and line (n) with the model named by SCNN_UNet_ConvLSTM2, could deliver better lane location predictions with thinner lines, compared with the baseline models. Thinner predicted lane lines mean that the model knows exactly where the lane should be rather than scattering the points around.

• The predicted lane lines should not merge or be broken. We can see in the 1st, 2nd, 6th, 7th, and 8th columns of Fig. 4, some baseline models' output lane lines either merge at the far end or broke the continuity with dashed lines. The proposed models perform slightly better although in a few cases the lines are also discontinuous.

•The lanes should be predicted correctly even at the boundary of the image. As can be found in Fig. 4, some baseline models, e.g., line (c), (d), and (e), have difficulties at the top boundary of the image with merge lanes on the top. This also accords with the above-aforementioned property.

• The lane detection models should deliver accurate predictions under different driving scenes, even under some challenging situations. For example, we can see in the 2nd, 3rd, 5th, and 7th columns of Fig. 4, there are vehicles occluding the lanes. A good lane detection model should be able to handle these. The proposed models perform well under these slightly challenging cases, more challenging situations are further discussed later.

2) tvtLane Testset #2: 12 challenging driving cases

Fig. 5 shows the comparison of the proposed models with the baseline models under some extremely challenging driving scenes in the tvtLane testset#2. All of the detection results are without postprocessing. These challenging scenes cover wide situations including serious vehicle occlusion, bad lighting conditions (e.g., shadow, dim), tunnel situations, and dirt road conditions. In some extreme cases, the whole lanes are covered by vehicles, other objects, and/or shadows, which could be very difficult even for humans to do the detection.

As can be observed in Fig. 5, although all the baseline models fail in these challenging cases, our proposed models, especially the one named SCNN_SegNet_ConvLSTM2 illustrated in row (k), could still deliver good predictions in almost every situation listed in Fig. 5. The only flaw is that in the 3rd column where vehicle occlusion and blur road conditions happen simultaneously, the proposed models also find it hard to predict precisely. With the results in the 4th, 7th, and 8th columns, the robustness of SCNN_SegNet_ConvLSTM2's property in detecting the correct number of lane lines is further verified, especially, we can observe in the 4th column, where almost all the other models are defeated, SCNN_SegNet_ConvLSTM2 can still predict the correct number of lanes.

Furthermore, we should notice correct lane location predictions in these challenging situations are of vital importance for safe driving. For example, regarding the situation in the last column where a heavy vehicle totally shadows the field of vision on the left side, it will be very dangerous if the automated

vehicle is driving according to the lane detection results demonstrated in the 3rd to 5th rows.

3.3 Quantitative evaluation

1) Evaluation metrics: We also examine the performance of the proposed methods with quantitative evaluations. When treated as a pixel-wise classification task, the most simple evaluation criterion for lane detection is the accuracy (Zou et al., 2017), which measures the overall classification performance based on correctly classified pixels, indicated in equation (11),

$$\text{Accuracy} = \frac{Truly\ Classified\ Pixels}{Total\ Number\ of\ Pixels} \tag{11}$$

However, since it is an imbalanced binary classification task, where the pixels that stand for the lanes are far less than those that stand for the background (generally the ratio between them is lower than 1/50), using only accuracy to evaluate the model is not suitable. Thus, we introduce Precision, Recall, and F-measure, illustrated by equation (12)-(14).

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive+False Positive}} \tag{12}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive+False Negative}} \tag{13}$$

$$\text{F-measure} = (1+\beta^2)\frac{\text{Precision}*\text{Recall}}{\beta^2\text{Precision+Recall}} \tag{14}$$

Here, true positive means the number of lane pixels that are correctly identified as lanes, false-positive represents the number of background pixels that are wrongly predicted as lanes, and false negative means the number of lane pixels that are wrongly predicted as background.

Specifically, we choose $\beta = 1$, which corresponds to the F1-measure (harmonic mean) shown in equation (15).

$$\text{F1-measure} = 2 * \frac{\text{Precision}*\text{Recall}}{\text{Precision+Recall}} \tag{15}$$

The F1-Measure, which balances Precision and Recall, always serves as the main benchmark for model evaluation, e.g., (Liu et al., 2021; Pan et al., 2018; Xu et al., 2020; Zhang et al., 2021; Zou et al., 2020).

2) Performance and comparisons on tvtLane testset #1(normal situations)

As shown in Table 2, the proposed model of SCNN_UNet_ConvLSTM2, performs the best when evaluating on tvtLane Testset#1, with the highest Accuracy and F1-Measure, while the proposed model of SCNN_SegNet_ConvLSTM2 delivers the best Precision.

Incorporating with the qualitative evaluation, we could easily interpret that the highest Precision, Accuracy, and F1-Measure are mainly due to the following properties of the model's lane prediction results: (a) the correct number of lanes, (b) the accurate position of each lane, (c) the continuity of the lane lines, and (d) the thinness of the predicted lane lines with less blurry, which accords with (b). The correct prediction directly reduces the number of False Positives, and a good Precision contributes to better Accuracy and F1-Measure.

With respect to the model's structure, a further explanation of the high Precision, Accuracy, and F1-Measure of our proposed model can be summarized as follows:

First, the SCNN layer embedding in the encoder equips our model with better information extracting ability regarding the low-level features and spatial relations in each image.

Second, the RNN blocks, i.e., ConvLSTM / ConvGRU layers, can effectively capture the temporal dependencies among the continuous image frames, which could be very helpful for challenging situations where the lanes are shadowed or covered by other objects in the current frame.

Finally, the proposed architecture could make full use of the spatial-temporal information by processing *K* frames and adjusting the relative weights of the convolutional kernels within the ConvLSTM / ConvGRU layers.

All in all, the proposed model tries to not only strengthen feature extraction regarding spatial relation in one image frame but also the temporal correlation and dependencies among image frames for lane detection.

Table 2.

Model performance comparison on tvtLane testset #1 (normal situations)

		Test_Acc (%)	Precision	Recall	F1-Measure	Params (M)
Models using **single image** as input	**Baseline Models**					
	U-Net	96.54	0.790	**0.985**	0.877	13.4
	SegNet	96.93	0.796	0.962	0.871	29.4
	SCNN*	96.79	0.654	0.808	0.722	19.2
	LaneNet*	97.94	0.875	0.927	0.901	19.7
Models using **continuous images sequence** as inputs	SegNet_ConvLSTM**	97.92	0.874	0.931	0.901	67.2
	UNet_ConvLSTM**	98.00	0.857	0.958	0.904	51.1
	Proposed Models (SegNet-Based)					
	SCNN_SegNet_ConvGRU1	98.00	0.878	0.935	0.905	43.7
	SCNN_SegNet_ConvGRU2	98.05	0.888	0.918	0.903	57.9
	SCNN_SegNet_ConvLSTM1	98.01	0.881	0.935	0.907	48.5
	SCNN_SegNet_ConvLSTM2	98.07	**0.893**	0.928	0.910	67.3
	Proposed Models (UNet-Based)					
	SCNN_UNet_ConvGRU1	98.13	0.878	0.957	0.916	27.7
	SCNN_UNet_ConvGRU2	**98.19**	0.887	0.950	0.917	41.9
	SCNN_UNet_ConvLSTM1	98.18	0.886	0.948	0.916	32.4
	SCNN_UNet_ConvLSTM2	**98.19**	0.889	0.950	**0.918**	51.3
	Proposed Models (Light Version UNet-Based)					
	SCNN_UNetLight_ConvGRU1	97.83	0.850	0.960	0.902	**6.9**
	SCNN_UNetLight_ConvGRU2	98.01	0.863	0.950	0.905	10.5
	SCNN_UNetLight_ConvLSTM1	97.71	0.830	0.950	0.886	8.1
	SCNN_UNetLight_ConvLSTM2	97.76	0.840	0.953	0.893	12.8

* Results reported in (Zhang et al., 2021).

** There are two hidden layers of ConvLSTM in SegNet_ConvLSTM and UNet_ConvLSTM.

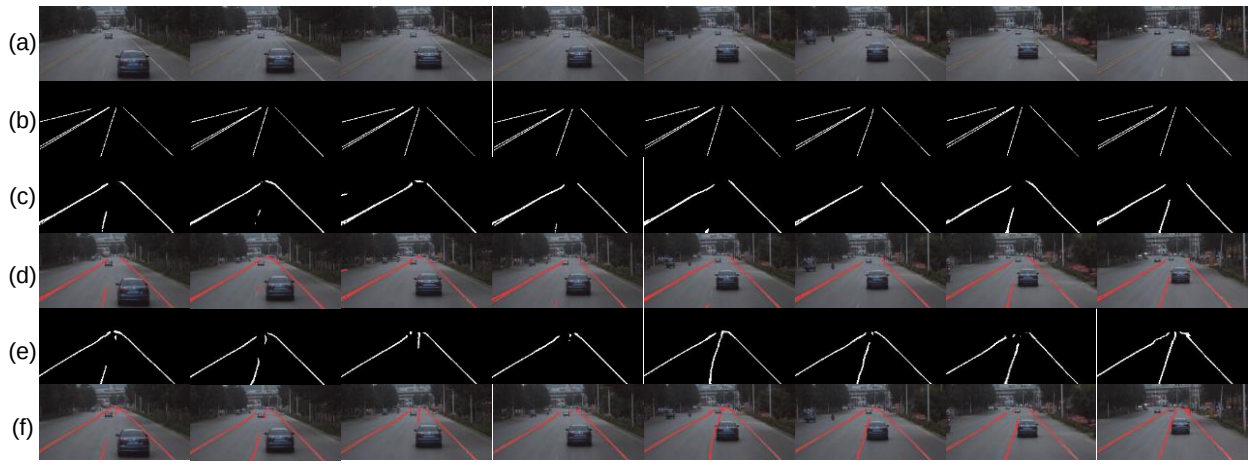
Table 3.
Model performance comparison on tvtLane testset #2 (12 types of challenging scenes)

PRECISION

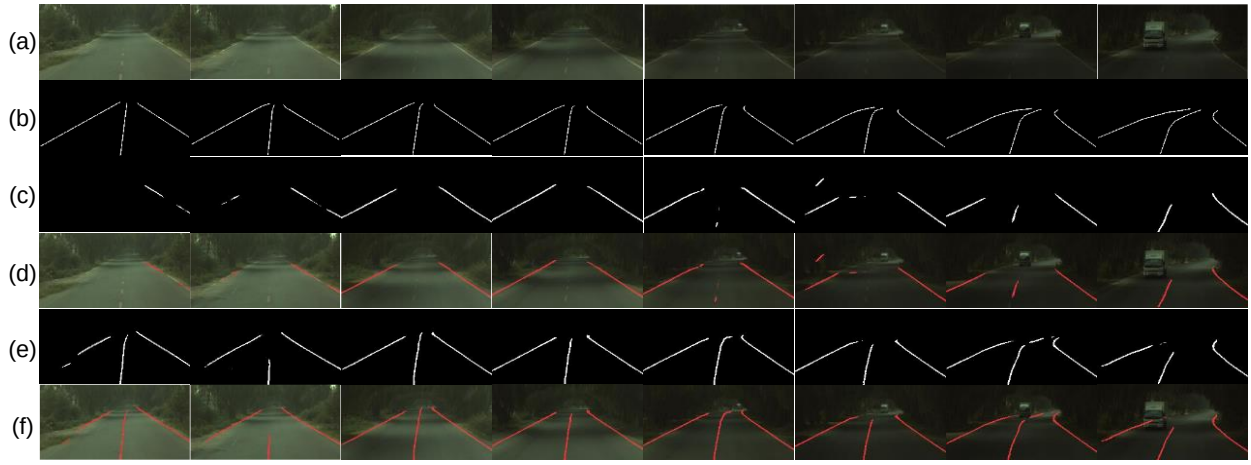
Challenging Scenes / Models	1-curve & occlude	2-shadow	3-bright	4-occlude	5-curve	6-dirty & occlude	7-urban	8-blur & curve	9-blur	10-shadow	11-tunnel	12-dim & occlude	overall
U-Net	0.7018	0.7441	0.6717	0.6517	0.7443	0.3994	0.4422	0.7612	0.8523	0.7881	0.7009	0.5968	0.6754
SegNet	0.6810	0.7067	0.5987	0.5132	0.7738	0.2431	0.3195	0.6642	0.7091	0.7499	0.6225	0.6463	0.6080
UNet_ConvLSTM	0.7591	0.8292	0.7971	0.6509	0.8845	0.4513	0.5148	**0.8290**	**0.9484**	**0.9358**	0.7926	0.8402	**0.7784**
SegNet_ConvLSTM	0.8176	0.8020	0.7200	0.6688	0.8645	0.5724	0.4861	0.7988	0.8378	0.8832	0.7733	0.8052	0.7563
SCNN_SegNet_ConvGRU1	0.8107	0.7951	0.7225	0.6830	0.8503	0.4640	0.5071	0.6699	0.8481	0.8994	0.7804	0.8429	0.7477
SCNN_SegNet_ConvGRU2	0.7952	0.8087	0.7770	0.6444	0.8689	0.5067	0.5171	0.7147	0.8423	0.8744	0.7979	**0.8757**	0.7572
SCNN_SegNet_ConvLSTM1	0.7945	0.8078	0.7600	0.6417	0.8525	0.5252	0.3686	0.7582	0.7715	0.8702	0.7778	0.8517	0.7348
SCNN_SegNet_ConvLSTM2	0.8326	0.7497	0.7470	0.7369	0.8647	**0.6196**	0.4333	0.7371	0.8566	0.9125	0.8153	0.8466	0.7673
SCNN_UNet_ConvGRU1	0.8492	0.8306	0.8163	**0.7845**	0.8819	0.4025	0.4493	0.7378	0.8291	0.8928	**0.8198**	0.8040	0.7639
SCNN_UNet_ConvGRU2	**0.8678**	0.7873	**0.8548**	0.7654	0.8805	0.5319	0.4735	0.8064	0.8765	0.8431	0.7112	0.7388	0.7640
SCNN_UNet_ConvLSTM1	0.8602	0.7844	0.8119	0.7807	**0.8871**	0.4066	0.4652	0.7445	0.8321	0.8972	0.7507	0.7068	0.7531
SCNN_UNet_ConvLSTM2	0.8182	**0.8362**	0.8189	0.7359	0.8365	0.5872	**0.5377**	0.8046	0.8770	0.8722	0.7952	0.7817	**0.7784**
SCNN_UNetLight_ConvGRU1	0.8212	0.7454	0.7189	0.6996	0.8521	0.3499	0.3999	0.7851	0.7282	0.8686	0.6940	0.6289	0.7011
SCNN_UNetLight_ConvGRU2	0.8147	0.8349	0.7390	0.7004	0.8591	0.4039	0.3360	0.6811	0.8300	0.8533	0.8125	0.7996	0.7238
SCNN_UNetLight_ConvLSTM1	0.7222	0.7450	0.6533	0.6203	0.8039	0.2635	0.2716	0.7341	0.7546	0.7319	0.6298	0.7406	0.6377
SCNN_UNetLight_ConvLSTM2	0.7618	0.7416	0.7067	0.6537	0.8096	0.1921	0.2639	0.6857	0.6830	0.6931	0.6391	0.6022	0.6190

F1-MEASURE

Challenging Scenes / Models	1-curve & occlude	2-shadow	3-bright	4-occlude	5-curve	6-dirty & occlude	7-urban	8-blur & curve	9-blur	10-shadow	11-tunnel	12-dim & occlude	overall
U-Net	0.8200	0.8408	0.7946	0.7337	0.7827	0.3698	0.5658	0.8147	0.7715	0.6619	0.5740	0.4646	0.6985
SegNet	0.8042	0.7900	0.7023	0.6127	0.8639	0.2110	0.4267	0.7396	0.7286	0.7675	0.6935	0.5822	0.6727
UNet_ConvLSTM	0.8465	**0.8891**	**0.8411**	0.7245	0.8662	0.2417	0.5682	0.8323	0.7852	0.6404	0.4741	0.5718	0.7143
SegNet_ConvLSTM	0.8852	0.8544	0.7688	0.6878	0.9069	0.4128	0.5317	0.7873	0.7575	0.8503	0.7865	0.7947	0.7609
SCNN_SegNet_ConvGRU1	0.8821	0.8626	0.7734	0.7185	0.9039	0.3027	0.5288	0.7229	**0.7866**	0.8658	0.7759	0.7763	0.7547
SCNN_SegNet_ConvGRU2	0.8710	0.8630	0.8094	0.6989	0.9005	0.3963	0.5497	0.7470	0.7637	0.8525	0.7798	0.7396	0.7591
SCNN_SegNet_ConvLSTM1	0.8768	0.8801	0.8185	0.7166	0.9083	0.3750	0.4516	0.7806	0.7320	0.8622	**0.8029**	**0.8245**	0.7629
SCNN_SegNet_ConvLSTM2	0.8956	0.8237	0.7909	0.7468	**0.9108**	**0.4398**	0.4858	0.7379	0.7546	**0.8729**	0.7963	0.8074	**0.7666**
SCNN_UNet_ConvGRU1	0.8608	0.8745	0.8393	**0.7802**	0.9005	0.3181	0.5143	0.7833	0.7567	0.5554	0.3503	0.3703	0.6839
SCNN_UNet_ConvGRU2	0.8706	0.8556	0.8304	0.7647	0.8532	0.3515	0.5253	**0.8345**	0.7399	0.5405	0.3567	0.2855	0.6722
SCNN_UNet_ConvLSTM1	**0.8971**	0.8493	0.8234	0.7633	0.8997	0.3054	0.5307	0.7424	0.7436	0.6243	0.5568	0.5366	0.6992
SCNN_UNet_ConvLSTM2	0.8670	0.8866	0.8405	0.7565	0.7955	0.4179	**0.5933**	0.7880	0.7285	0.6296	0.4747	0.4134	0.7024
SCNN_UNetLight_ConvGRU1	0.8896	0.8212	0.7819	0.7517	0.8913	0.3043	0.4961	0.8133	0.7000	0.5635	0.3086	0.2733	0.6637
SCNN_UNetLight_ConvGRU2	0.8593	0.8730	0.7878	0.7406	0.8889	0.3335	0.4266	0.7263	0.7782	0.6498	0.5280	0.5257	0.6910
SCNN_UNetLight_ConvLSTM1	0.8115	0.8056	0.7168	0.6882	0.8179	0.2613	0.3681	0.7834	0.7576	0.5701	0.5281	0.5081	0.6418
SCNN_UNetLight_ConvLSTM2	0.8377	0.8158	0.7620	0.6971	0.8365	0.2209	0.3577	0.7551	0.6594	0.4597	0.3545	0.3559	0.6079



(1) Challenging situation **8-blur&curve**



(2) Challenging situation **10-shadow**

Fig. 6. Visual comparison of the lane-detection results on challenging driving situations for UNet_ConvLSTM and the proposed model SCNN_SegNet_ConvLSTM2. All of the detection results are without postprocessing. (a) Input images. (b) Ground truth. (c) Lane detection results of UNet_ConvLSTM. (d) Lane detection results of UNet_ConvLSTM overlapping on the original images. (e) Lane detection results of SCNN_SegNet_ConvLSTM2. (f) Lane detection results of SCNN_SegNet_ConvLSTM2 overlapping on the original images. The upper part (1) is for challenging situation **8-blur&curve**, while the down part (2) is for situation **10-shadow**.

Looking at the main metric, F1-Measure, experiments show that it is impossible to increase the F1-Measure by increasing only the Precision or only Recall. Thus we can observe that although U-Net, SegNet, and SegNet_ConvLSTM get the higher Recalls, they do not deliver good F1-Measures since their Precision is much lower than the proposed model of SCNN_SegNet_ConvLSTM2 and SCNN_UNet_ConvLSTM2. Regarding the good Recall of U-Net and SegNet, it could be speculated from the qualitative evaluation, where we can find that U-Net and SegNet tend to produce thicker lane lines. With thicker lines and blurry areas, the two models can somehow reduce the False Negative, which will contribute to better Recall. This also demonstrates that Recall and Precision antagonize each other which further proves that F1-Measure should be a more reasonable evaluation measure compared with Precision and Recall.

3) Performance and comparisons on tvtLane testset #2 (challenging situations)

To further evaluate the proposed models' performance and verify the models' robustness, we test the models on a brand-new and dataset, i.e., the tvtLane Testset #2. As introduced in ***3.1 Datasets***, tvtLane

Testset #2 contains 728 images including lanes in rural, urban, and highway scenes. These challenging driving scenes' data are captured by data recorders at different heights, inside and outside the front windshield, and with different weather and road conditions. It is a comprehensive and challenging test dataset in which some cases are difficult enough, even for human eyes, to do the correct detection.

Table 3 shows the model performance comparison on the 12 types of challenging scenes in tvtLane Testset #2. Following the results and discussions in *2) Performance and comparisons on tvtLane testset #1(normal situations)*, here we provide the Precision and F1-Measure for evaluation reference.

As indicated by the bold numbers, our proposed model, SCNN_SegNet_ConvLSTM2, delivers the best F1-Measure at the overall level and in more situations, while UNet_ConvLSTM gets the best Precision at the overall level and in more situations. Incorporating with the qualitative evaluation in Fig. 5, we find that UNet_ConvLSTM tends to not classify pixels into lane lines for uncertain areas under some challenging situations (e.g., the 2nd and 7th columns in Fig. 5). This might be the reason for its obtaining better Precision. To further confirm this speculation, in Fig. 6 we compared the lane detection results of SCNN_SegNet_ConvLSTM2 and UNet_ConvLSTM under challenging situations **8-blur&curve**, and **10-shadow**, where UNet_ConvLSTM delivers very good Precisions.

As illustrated in Fig. 6, truly UNet_ConvLSTM tries not to classify pixels into lane lines under uncertain areas as much as possible. This leads to fewer False Negatives which helps for raising a better Precision. However, in real application scenarios, this is not wise and not acceptable. On the contrary, our proposed model SCNN_SegNet_ConvLSTM2 tries to make tough but valuable detections classifying candidate points into lane lines in the challenging uncertain areas with dirt, dark road conditions, and/or vehicle occlusions. This may lead to more False Negatives and a worse Precision but is praiseworthy. These analyses further demonstrate that F1-Measure is a better measure compared with Precision. Finally, we can conclude that our proposed model, SCNN_SegNet_ConvLSTM2, delivers the best performance on the challenging tvtLane Testset #2, which verified the proposed model architecture's robustness.

To sum up, the proposed model architecture demonstrates its effectiveness in both normal and challenging driving scenes, with the UNet based model, SCNN_UNet_ConvLSTM2, beats the baseline models with a large margin on normal situations, while the SegNet based model, SCNN_SegNet_ConvLSTM2 performs the best handling the challenging driving scenes. The finding that, compared with UNet based models, SegNet based neural network models are more robust coping with challenging driving environments accords with results in (Zou et al., 2020).

3.4 Parameter analysis and ablation study

1) The added value of SCNN

Regarding the neural network architecture, we investigate the effects of the model with and without SCNN layers. As demonstrated in Fig. 4 and Fig. 5, together with the quantitative results in Table 2 and Table 3, the proposed SegNet and UNet based models with SCNN embedded encoder, i.e., SCNN_SegNet_ConvLSTM, SCNN_SegNet_ConvGRU, SCNN_UNet_ConvLSTM, and SCNN_UNet_ConvGRU, outperform SegNet_ConvLSTM and UNet_ConvLSTM which are also SegNet or UNet based sequential model using multiple continuous image frames as inputs. Especially,

SCNN_UNet_ConvLSTM2 obtains the best result in normal testing while SCNN_SegNet_ConvLSTM2 delivers the best performance in challenging situations.

For normal cases' testing on tvtLane Testset#1, in Table 2, we can observe, by adding SCNN layer in the encoder, almost all the proposed models with SCNN embedded encoder outperform the baseline models with better F1-Measure. To be specific, SCNN_SegNet_ConvLSTM2 improves the lane detection accuracy by around 0.3% and F1-measure by around 1%, and these improvements are from the already very good results obtained by SegNet_ConvLSTM. Similarly, SCNN_UNet_ConvLSTM2 overperforms UNet_ConvLSTM with even larger margins regarding both Accuracy, Precision, and F1-measure.

For challenging situations, adding SCNN layer also helps the proposed model, SCNN_SegNet_ConvLSTM2, beat other baseline models, and deliver the best F1-Measure as indicated in Table 3.

Fig. 7 visualizes the extracted low-level features at *Down_ConvBlock_1* layer for UNet based models, with and without SCNN. Clearly, vast differences can be witnessed between the baseline model UNet_ConvLSTM and the proposed model SCNN_UNet_ConvLSTM2. In Fig. 7 (a), the CNN-based UNet layers identify the low-level features in the images regarding the target lane lines. However, the extracted features are not so clear, i.e., there are some interference signals, especially as visualized in the third image of row (a), which is supposed to affect the model training (i.e., updating weight parameters of the neural networks) and thus affect the model's performance regarding detection results. It may further affect the generation of the final lane detection results. In contrast, the extracted low-level features of the lane lines are more inerratic, clear, and evident as shown in Fig. 7 (b). There are fewer interferences surrounding the detected lane features. This verifies SCNN's powerful strength in detecting the spatial relations in every single image with its message passing mechanism.

All the above results demonstrate the adding of the SCNN layer embedded in the encoder does contribute to the spatial feature extraction, with which the model could better make full use of the spatial-temporal information among the image frames.

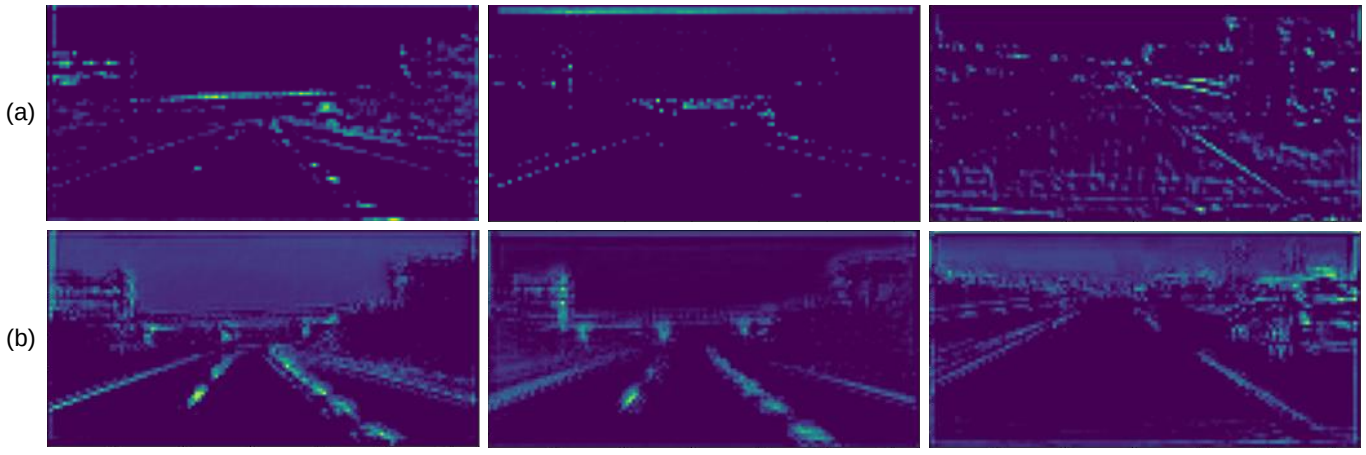


Fig. 7. Visualization of the extracted low-level features at Down_ConvBlock_1 for UNet based models. (a) Results of UNet_ConvLSTM (without SCNN layers). (b) Results of the SCNN_UNet_ConvLSTM2 (with SCNN layers).

2) Different locations of SCNN layer

We also investigated the effects of different locations of the SCNN layer. The experimental results of testing different locations of the SCNN layer in our proposed model are shown in Table 4. The results reveal that: (a) Compared with baseline models without SCNN layers, the embedding of SCNN layers really help to improve the models' performance, which further verifies the added-value of SCNN and accords with the aforementioned results in ***1)***; (b) In terms of the main evaluation metric F1-measure, embedding SCNN layer after the *Conv1-1* (in SegNet based model) or *In_Conv-1* (in UNet based model) layer delivers better results compared with embedding it at the very beginning or early layers of the encoder; (c) For UNet based model, embedding SCNN layer at the very beginning delivers quite good Precision and Accuracy, but worse Recall, which means there are fewer False Positives but more False Negatives. This should be related to the properties of the UNet style neural network. These results further confirm the effectiveness of the proposed model architecture.

Table 4.

Model performance comparison different location of SCNN layer on tvtLane testset #1 and testset #2

Models \ Testing Datasets		Testset #1 (Normal Situations)				Testset #2 (Challenging Scenes)			
	Location of SCNN	Test_Acc (%)	Precision	Recall	F1-Measure	Test_Acc (%)	Precision	Recall	F1-Measure
SegNet_ConvLSTM	Without	97.92	0.874	0.931	0.901	97.83	0.756	0.765	0.761
SCNN_SegNet_ConvLSTM2	Conv1-1	98.00	0.884	0.921	0.902	97.92	0.757	0.757	0.757
	Conv2-1	98.07	0.893	0.928	0.910	97.90	0.767	0.766	0.767
UNet_ConvLSTM	Without	98.00	0.857	0.957	0.904	97.93	0.778	0.660	0.714
SCNN_UNet_ConvLSTM2	In_Conv-1	98.28	0.896	0.939	0.917	98.08	0.776	0.593	0.672
	Conv1-1	98.19	0.889	0.950	0.918	97.95	0.778	0.640	0.702

3) Type and number of RNN layers

As described in *Section 3*, in our proposed model architecture we employed two types of RNNs, i.e., ConvLSTM and ConvGRU, serving as the RNN block, to capture and make use of the spatial-temporal correlations and time dependencies among the continuous image sequences. We also tested the number of hidden ConvLSTM and ConvGRU layers from 1 to 2. The quantitative results are demonstrated in Table 2 and Table 3, while some intuitive qualitative insights could be drawn from Fig. 4. and Fig. 5.

From Table 2, we can see that in general models adopting ConvLSTM layers in the RNN block perform better than those adopting ConvGRU layers with improved F1-measure, except for the UNetLight based models which will be discussed below in ***4)***. This could be explained by ConvLSTM's better properties in extracting spatial-temporal features and capturing time dependencies by more control gates and thus more parameters compared with ConvGRU. Furthermore, from Table 2 and Table 3, we can see that models with two hidden RNN layers, for both ConvLSTM and ConvGRU, generally perform better than those with only one hidden RNN layer. This could be speculated that with two hidden RNN layer, one layer can serve for sequential feature extraction, and the other can achieve spatial-temporal feature integration. The improvements of two RNN layers over one are not that

significant which might be due to (a) models employing one RNN layer already obtain good results; (b) since the length of the continuous image frames is only five, one RNN layer might be already enough to do the spatial-temporal feature extraction, so when incorporating longer image sequences the superiorities of two RNN layers could be promoted. However, longer image sequences require more computational resources and longer training time, which we could not afford at the present stage. This could be the future research direction.

4) Number of parameters and real-time capability

As shown in Table 2, the two proposed candidate models, i.e., SCNN_SegNet_ConvLSTM2 and SCNN_UNet_ConvLSTM2, possess a bit more parameters compared with the baseline SegNet_ConvLSTM and UNet_ConvLSTM, respectively. However, almost all of our proposed model variants with different types and numbers of RNN layers outperform the baselines, and some of them are even with low parameter sizes e.g., SCNN_SegNet_ConvGRU1, SCNN_SegNet_ConvLSTM1, SCNN_UNet_ConvGRU1, SCNN_UNet_ConvLSTM1. Generally speaking, lower numbers of model parameters mean better real-time capability.

In addition, to reduce the total parameter size and improve the model's ability to operate in real-time, we also implemented four model variants with a modified light version of UNet, i.e., UNetLight, serving as the network backbone. The UNetLight backbone has a similar network design with UNet whose parameter settings are demonstrated in Table A2. The only difference is that all the numbers of kernels in the ConvBlocks are reduced to half except for the *Input* in *In_ConvBlock* (with the input channel of 3 unchanged) and *Output* in Out_*ConvBlock* (with the output channel of 2 unchanged). From the testing results in Table 2, we can find that the model named SCNN_UNetLight_ConvGRU2, with fewer parameters than all the baseline models, beat the baselines exhibiting better performance regarding both Accuracy and F1-Measure. To be specific, compared with the best baseline model i.e., UNet_ConvLSTM, SCNN_UNetLight_ConvGRU2 only uses no more than one-fifth of the parameter size but delivers better evaluation metrics in testing Accuracy, Precision, and F1-Measure.

Regarding UNetLight based models, we can also discover that models using ConvGRU layers as the RNN block performs better than those adopting ConvLSTM. The reason could be that light version UNet can not implement high-quality feature extraction which does not feed enough information for ConvLSTM, while ConvGRU, with fewer control gates, is more robust when low-level features are not that fully extracted.

All these results further verify the proposed network architecture's effectiveness and strength.

4. Conclusion

In this paper, a novel spatial-temporal sequence-to-one model with a hybrid neural network architecture integrating encoder-decoder structure together with SCNN and ConvLSTM/ ConvGRU is proposed for robust lane detection under various normal and challenging driving scenes. Compared with baseline models which use one single image (e.g., U-Net, SegNet, and LaneNet), as well as the state-of-art models adopting "CNN+RNN" structures (e.g., UNet_ConvLSTM, SegNet_ConvLSTM), the proposed architecture achieved significantly better results, with the best testing Accuracy, Precision, F1-measure on the normal driving dataset (i.e., *tvtLane* Testset #1) and the best F1-measure on the 12

challenging driving scenarios dataset (*tvtLane* Testset #2). The results verify the effectiveness of strengthing spatial relation abstraction in every single image with SCNN layer, plus the employing of multiple continuous image sequences as inputs, and also verify the proposed model architecture's ability in making full use of the spatial-temporal information in continuous image frames. Extensive experimental results also demonstrated the superiorities of the sequence-to-one "SCNN + ConvLSTM" over ordinary "CNN + ConvLSTM" and "SCNN + ConvGRU" regarding sequential spatial-temporal feature learning and target-information prediction in the context of robust lane detection. Besides, testing results of the model variants with the modified light version of UNet (i.e., UNetLight) as the backbone demonstrate the proposed model architecture's potential regarding real-time capability.

To the best of the authors' knowledge, the proposed model is the first attempt that tries to strengthen both spatial relations regarding feature extraction in every image frame together with the temporal correlations and dependencies among image frames for lane detection, and the extensive evaluation experiments demonstrate the strength of this proposed architecture. We suggest future studies should also incorporate both aspects to obtain better performance.

In this paper, the challenging cases do not include night driving, rainy or wet road conditions. There are demands to build larger test sets with comprehensive challenging situations. Since we had collected a large amount of unlabeled driving scene data dedicated for lane detection, a future research direction might be to develop semi-supervised learning methods and employ domain adaption to label the collected data, and then open source them for boosting the research in the field of robust lane detection. We also plan to further enhance the lane detection model by introducing the sequential attention mechanism into the proposed framework.

CRediT authorship contribution statement

Yongqi Dong: Conceptualization, Investigation, Methodology, Data curation, Visualization, Validation, Writing - original draft, Writing - review & editing. **Sandeep Patil**: Data curation, Visualization. **Bart van Arem**: Supervision, Validation, Resources, Writing - review & editing. **Haneen Farah**: Supervision, Validation, Resources, Writing - review & editing, Project administration, Funding acquisition.

Acknowledgment

This work was supported by the Applied and Technical Sciences (TTW), a subdomain of the Dutch Institute for Scientific Research (NWO) through the Project Safe and Efficient Operation of Automated and Human-Driven Vehicles in Mixed Traffic (SAMEN) under Contract 17187. The authors thank Dr. Qin Zou, Hanwen Jiang, and Qiyu Dai from Wuhan University, as well as Jiyong Zhang from Southwest Jiaotong University for their tips in using the tvtLane dataset.

Appendix A.

See Table A1 and Table A2.

Table A1.

SegNet-based network architecture parameters.

Layer		Input (channel×hight×width)	Output (channel×hight×width)	Kernel	Stride	Pad	Activation
Down_ConvBlock_1	Conv1-1	3×128×256	64×128×256	3×3	1	(1,1)	ReLU
	Conv1-2	64×128×256	64×128×256	3×3	1	(1,1)	ReLU
	Maxpool1	64×128×256	64×64×128	2×2	2	(0,0)	---
SCNN	SCNN_Down	64×1×128	64×1×128	1×9	1	(0,4)	ReLU
	SCNN_Up	64×1×128	64×1×128	1×9	1	(0,4)	ReLU
	SCNN_Right	64×64×1	64×64×1	9×1	1	(4,0)	ReLU
	SCNN_Left	64×64×1	64×64×1	9×1	1	(4,0)	ReLU
Down_ConvBlock_2	Conv2-1	64×64×128	128×64×128	3×3	1	(1,1)	ReLU
	Conv2-2	128×64×128	128×64×128	3×3	1	(1,1)	ReLU
	Maxpool2	128×64×128	128×32×64	2×2	2	(0,0)	---
Down_ConvBlock_3	Conv3-1	128×32×64	256×32×64	3×3	1	(1,1)	ReLU
	Conv3-2	256×32×64	256×32×64	3×3	1	(1,1)	ReLU
	Conv3-3	256×32×64	256×32×64	3×3	1	(1,1)	ReLU
	Maxpool3	256×64×128	256×16×32	2×2	2	(0,0)	---
Down_ConvBlock_4	Conv4-1	256×16×32	512×16×32	3×3	1	(1,1)	ReLU
	Conv4-2	512×16×32	512×16×32	3×3	1	(1,1)	ReLU
	Conv4-3	512×16×32	512×16×32	3×3	1	(1,1)	ReLU
	Maxpool4	512×16×32	512×8×16	2×2	2	(0,0)	---
Down_ConvBlock_5	Conv5-1	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
	Conv5-2	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
	Conv5-3	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
	Maxpool5	512×8×16	512×4×8	2×2	2	(0,0)	---
RNN Layer1*	5 * ConvLSTMCell(input=(512×4×8), kernel=(3,3), stride=(1,1), padding=(1,1)) ***Or*** 5 * ConvGRUCell(input=(512×4×8), kernel=(3,3), stride=(1,1), padding=(1,1), dropout(0.5))						
RNN Layer2**	5 * ConvLSTMCell(input=(512×4×8), kernel=(3,3), stride=(1,1), padding=(1,1)) ***Or*** 5 * ConvGRUCell(input=(512×4×8), kernel=(3,3), stride=(1,1), padding=(1,1), dropout(0.5))						
Up_ConvBlock_5	MaxUnpool1	512×4×8	512×8×16	2×2	2	(0,0)	---
	Up_Conv5-1	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
	Up_Conv5-2	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
	Up_Conv5-3	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
Up_ConvBlock_4	MaxUnpool2	512×8×16	512×16×32	2×2	2	(0,0)	---
	Up_Conv4-1	512×16×32	512×16×32	3×3	1	(1,1)	ReLU
	Up_Conv4-2	512×16×32	512×16×32	3×3	1	(1,1)	ReLU
	Up_Conv4-3	512×16×32	256×16×32	3×3	1	(1,1)	ReLU
Up_ConvBlock_3	MaxUnpool3	256×16×32	256×32×64	2×2	2	(0,0)	---
	Up_Conv3-1	256×32×64	256×32×64	3×3	1	(1,1)	ReLU
	Up_Conv3-2	256×32×64	256×32×64	3×3	1	(1,1)	ReLU
	Up_Conv3-3	256×32×64	128×32×64	3×3	1	(1,1)	ReLU
Up_ConvBlock_2	MaxUnpool4	128×32×64	128×64×128	2×2	2	(0,0)	---
	Up_Conv2-1	128×64×128	128×64×128	3×3	1	(1,1)	ReLU
	Up_Conv2-2	128×64×128	64×64×128	3×3	1	(1,1)	ReLU
Up_ConvBlock_1	MaxUnpool5	64×64×128	64×128×256	2×2	2	(0,0)	---
	Up_Conv1-1	64×128×256	64×128×256	3×3	1	(1,1)	ReLU
	Up_Conv1-2	64×128×256	2×128×256	3×3	1	(1,1)	LogSoftmax

* We test two types of RNN layers, i.e., ConvLSTM and ConvGRU;

** We test the RNN blocks with 1 hidden layer or 2 hidden layers.

Table A2.

UNet-based network architecture parameters

Layer		Input (channel×hight×width)	Output (channel×hight×width)	Kernel	Stride	Pad	Activation
In_ConvBlock	In_Conv-1	3×128×256	64×128×256	3×3	1	(1,1)	ReLU
	In_Conv-2	64×128×256	64×128×256	3×3	1	(1,1)	ReLU
SCNN	SCNN_Down	64×1×256	64×1×256	1×9	1	(0,4)	ReLU
	SCNN_Up	64×1×256	64×1×256	1×9	1	(0,4)	ReLU
	SCNN_Right	64×128×1	64×128×1	9×1	1	(4,0)	ReLU
	SCNN_Left	64×128×1	64×128×1	9×1	1	(4,0)	ReLU
Down_ConvBlock_1	Maxpool1	64×128×256	64×64×128	2×2	2	(0,0)	---
	Conv1-1	64×64×128	128×64×128	3×3	1	(1,1)	ReLU
	Conv1-2	128×64×128	128×64×128	3×3	1	(1,1)	ReLU
Down_ConvBlock_2	Maxpool2	128×64×128	128×32×64	2×2	2	(0,0)	---
	Conv2-1	128×32×64	256×32×64	3×3	1	(1,1)	ReLU
	Conv2-2	256×32×64	256×32×64	3×3	1	(1,1)	ReLU
Down_ConvBlock_3	Maxpool3	256×32×64	256×16×32	2×2	2	(0,0)	---
	Conv3-1	256×16×32	512×16×32	3×3	1	(1,1)	ReLU
	Conv3-2	512×16×32	512×16×32	3×3	1	(1,1)	ReLU
Down_ConvBlock_4	Maxpool4	512×16×32	512×8×16	2×2	2	(0,0)	---
	Conv4-1	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
	Conv4-2	512×8×16	512×8×16	3×3	1	(1,1)	ReLU
RNN Layer1*	5 * ConvLSTMCell(input=(512×8×16), kernel=(3,3), stride=(1,1), padding=(1,1)) ***Or*** 5 * ConvGRUCell(input=(512×8×16), kernel=(3,3), stride=(1,1), padding=(1,1), dropout(0.5))						
RNN Layer2**	5 * ConvLSTMCell(input=(512×8×16), kernel=(3,3), stride=(1,1), padding=(1,1)) ***Or*** 5 * ConvGRUCell(input=(512×8×16), kernel=(3,3), stride=(1,1), padding=(1,1), dropout(0.5))						
Up_ConvBlock_4	UpsamplingBilinear2D-1	512×8×16	512×16×32	2×2	2	(0,0)	---
	Up_Conv5-1	1024×16×32	256×16×32	3×3	1	(1,1)	ReLU
	Up_Conv5-2	256×16×32	256×16×32	3×3	1	(1,1)	ReLU
Up_ConvBlock_3	UpsamplingBilinear2D-2	256×16×32	256×32×64	2×2	2	(0,0)	---
	Up_Conv4-1	512×32×64	128×32×64	3×3	1	(1,1)	ReLU
	Up_Conv4-2	128×32×64	128×32×64	3×3	1	(1,1)	ReLU
Up_ConvBlock_2	MaxUnpool3	128×32×64	128×64×128	2×2	2	(0,0)	---
	Up_Conv3-1	156×64×128	64×64×128	3×3	1	(1,1)	ReLU
	Up_Conv3-2	64×64×128	64×64×128	3×3	1	(1,1)	ReLU
Up_ConvBlock_1	MaxUnpool4	64×64×128	64×128×256	2×2	2	(0,0)	---
	Up_Conv2-1	128×128×256	64×128×256	3×3	1	(1,1)	ReLU
	Up_Conv2-2	64×128×256	64×128×256	3×3	1	(1,1)	ReLU
Out_ConvBlock	Out_Conv	64×128×256	2×128×256	1×1	1	(0,0)	---

* Similar to the SegNet-based network architecture, we test two types of RNN layers, i.e., ConvLSTM and ConvGRU;

** We test the RNN blocks with 1 hidden layer or 2 hidden layers.

References

Aly, M., 2008. Real time detection of lane markers in urban streets. IEEE Intell. Veh. Symp. Proc. 7–12. https://doi.org/10.1109/IVS.2008.4621152

Andrade, D.C., Bueno, F., Franco, F.R., Silva, R.A., Neme, J.H.Z., Margraf, E., Omoto, W.T., Farinelli, F.A., Tusset, A.M., Okida, S., Santos, M.M.D., Ventura, A., Carvalho, S., Amaral, R.D.S., 2019. A Novel Strategy for Road Lane Detection and Tracking Based on a Vehicle's Forward Monocular Camera. IEEE Trans. Intell. Transp. Syst. 20, 1497–1507. https://doi.org/10.1109/TITS.2018.2856361

Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39, 2481–2495.

https://doi.org/10.1109/TPAMI.2016.2644615
Ballas, N., Yao, L., Pal, C., Courville, A., 2016. Delving deeper into convolutional networks for learning video representations, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings.
Bar Hillel, A., Lerner, R., Levi, D., Raz, G., 2014. Recent progress in road and lane detection: A survey. Mach. Vis. Appl. 25, 727–745. https://doi.org/10.1007/s00138-011-0404-2
Berriel, R.F., de Aguiar, E., de Souza, A.F., Oliveira-Santos, T., 2017. Ego-Lane Analysis System (ELAS): Dataset and algorithms. Image Vis. Comput. https://doi.org/10.1016/j.imavis.2017.07.005
Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers. https://doi.org/10.1007/978-3-7908-2604-3_16
Chen, W., Wang, W., Wang, K., Li, Z., Li, H., Liu, S., 2020. Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation–a review. J. Traffic Transp. Eng. (English Ed. https://doi.org/10.1016/j.jtte.2020.10.002
Chen, Z., Liu, Q., Lian, C., 2019. PointLaneNet: Efficient end-to-end CNNs for accurate real-time lane detection. IEEE Intell. Veh. Symp. Proc. 2019-June, 2563–2568. https://doi.org/10.1109/IVS.2019.8813778
Choi, Y., Park, J.H., Jung, H., 2018. Lane Detection Using Labeling Based RANSAC Algorithm 3, 245–248.
Du, H., Xu, Z., Ding, Y., 2018. The fast lane detection of road using RANSAC algorithm, in: Advances in Intelligent Systems and Computing. https://doi.org/10.1007/978-3-319-67071-3_1
Guo, J., Wei, Z., Miao, D., 2015. Lane Detection Method Based on Improved RANSAC Algorithm, in: Proceedings - 2015 IEEE 12th International Symposium on Autonomous Decentralized Systems, ISADS 2015. https://doi.org/10.1109/ISADS.2015.24
Haris, M., Glowacz, A., 2021. Lane line detection based on object feature distillation. Electron. 10. https://doi.org/10.3390/electronics10091102
Hochreiter, S., Schmidhuber, J., 1997. Long Short Term Memory. Neural Computation. Neural Comput.
Hou, Y., Ma, Z., Liu, C., Hui, T.W., Loy, C.C., 2020. Inter-Region Affinity Distillation for Road Marking Segmentation. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 12483–125492. https://doi.org/10.1109/CVPR42600.2020.01250
Jiao, X., Yang, D., Jiang, K., Yu, C., Wen, T., Yan, R., 2019. Real-time lane detection and tracking for autonomous vehicle applications. Proc. Inst. Mech. Eng. Part D J. Automob. Eng. https://doi.org/10.1177/0954407019866989
Kim, J., Park, C., 2017. End-To-End Ego Lane Estimation Based on Sequential Transfer Learning for Self-Driving Cars, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. https://doi.org/10.1109/CVPRW.2017.158
Ko, Y., Lee, Y., Azam, S., Munir, F., Jeon, M., Pedrycz, W., 2020. Key Points Estimation and Point Instance Segmentation Approach for Lane Detection 1–10. https://doi.org/10.1109/tits.2021.3088488
Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J.Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., Thrun, S., 2011. Towards fully autonomous driving: Systems and algorithms, in: IEEE Intelligent Vehicles Symposium, Proceedings. https://doi.org/10.1109/IVS.2011.5940562
Li, X., Li, J., Hu, X., Yang, J., 2020. Line-CNN: End-to-End Traffic Line Detection with Line Proposal Unit. IEEE Trans. Intell. Transp. Syst. 21, 248–258. https://doi.org/10.1109/TITS.2019.2890870
Liang, D., Guo, Y.C., Zhang, S.K., Mu, T.J., Huang, X., 2020. Lane Detection: A Survey with New Results. J. Comput. Sci. Technol. 35, 493–505. https://doi.org/10.1007/s11390-020-0476-4
Liu, L., Chen, X., Zhu, S., Tan, P., 2021. CondLaneNet: a Top-to-down Lane Detection Framework Based on Conditional Convolution.
Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J., 2019. On the Variance of the Adaptive Learning Rate and Beyond 1–14.
Liu, R., Yuan, Z., Liu, T., Xiong, Z., 2020. End-to-end Lane Shape Prediction with Transformers 3694–3702. https://doi.org/10.1109/wacv48630.2021.00374
Liu, T., Chen, Z., Yang, Y., Wu, Z., Li, H., 2020. Lane Detection in Low-light Conditions Using an Efficient Data Enhancement : Light Conditions Style Transfer.
Lu, Z., Xu, Y., Shan, X., Liu, L., Wang, X., Shen, J., 2019. A lane detection method based on a ridge detector and regional G-RANSAC. Sensors (Switzerland). https://doi.org/10.3390/s19184028

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L., 2018. Towards End-to-End Lane Detection: An Instance Segmentation Approach. IEEE Intell. Veh. Symp. Proc. 2018-June, 286–291. https://doi.org/10.1109/IVS.2018.8500547

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L., 2017. Fast Scene Understanding for Autonomous Driving.

Pan, X., Shi, J., Luo, P., Wang, X., Tang, X., 2018. Spatial as deep: Spatial CNN for traffic scene understanding, in: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018. AAAI press, pp. 7276–7283.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks, in: 30th International Conference on Machine Learning, ICML 2013.

Philion, J., 2019. FastDraw: Addressing the long tail of lane detection by adapting a sequential prediction network. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2019-June, 11574–11583. https://doi.org/10.1109/CVPR.2019.01185

Qin, Z., Wang, H., Li, X., 2020. Ultra Fast Structure-aware Deep Lane Detection.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 9351, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Schön, T.B., 2020. Using Attractors and Smoothness.

Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.C., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems.

Sivaraman, S., Trivedi, M.M., 2013. Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. IEEE Trans. Intell. Transp. Syst. 14, 906–917. https://doi.org/10.1109/TITS.2013.2246835

Sutskever, I., Vinyals, O., Le, Q. V., 2014. Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems.

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

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

Wang, B.F., Qi, Z.Q., Ma, G.C., 2014. Robust lane recognition for structured road based on monocular vision. J. Beijing Inst. Technol. (English Ed. 23, 345–351.

Wang, S., Hou, X., Zhao, X., 2020. Automatic Building Extraction from High-Resolution Aerial Imagery via Fully Convolutional Encoder-Decoder Network with Non-Local Block. IEEE Access. https://doi.org/10.1109/ACCESS.2020.2964043

Wang, Y., Dahnoun, N., Achim, A., 2012. A novel system for robust lane detection and tracking. Signal Processing. https://doi.org/10.1016/j.sigpro.2011.07.019

Wu, B., Li, K., Ge, F., Huang, Z., Yang, M., Siniscalchi, S.M., Lee, C.H.L., 2017. An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic modeling for robust speech recognition. IEEE J. Sel. Top. Signal Process. https://doi.org/10.1109/JSTSP.2017.2756439

Xing, Y., Lv, C., Chen, L., Wang, Huaji, Wang, Hong, Cao, D., Velenis, E., Wang, F.Y., 2018. Advances in Vision-Based Lane Detection: Algorithms, Integration, Assessment, and Perspectives on ACP-Based Parallel Vision. IEEE/CAA J. Autom. Sin. 5, 645–661. https://doi.org/10.1109/JAS.2018.7511063

Xu, H., Wang, S., Cai, X., Zhang, W., Liang, X., Li, Z., 2020. CurveLane-NAS: Unifying Lane-Sensitive Architecture Search and Adaptive Point Blending 1–16.

Yasrab, R., Gu, N., Zhang, X., 2017. An encoder-decoder based Convolution Neural Network (CNN) for future Advanced Driver Assistance System (ADAS). Appl. Sci. https://doi.org/10.3390/app7040312

Yoo, S., Seok Lee, H., Myeong, H., Yun, S., Park, H., Cho, J., Hoon Kim, D., 2020. End-to-end lane marker detection via row-wise classification. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work. 2020-June, 4335–4343. https://doi.org/10.1109/CVPRW50498.2020.00511

Zhang, J., Deng, T., Yan, F., Liu, W., 2021. Lane Detection Model Based on Spatio-Temporal Network With Double Convolutional Gated Recurrent Units. IEEE Trans. Intell. Transp. Syst. https://doi.org/10.1109/TITS.2021.3060258

Zheng, F., Luo, S., Song, K., Yan, C.W., Wang, M.C., 2018. Improved Lane Line Detection Algorithm Based on Hough Transform. Pattern Recognit. Image Anal. 28, 254–260.

https://doi.org/10.1134/S1054661818020049
Zou, Q., Jiang, H., Dai, Q., Yue, Y., Chen, L., Wang, Q., 2020. Robust lane detection from continuous driving scenes using deep neural networks. IEEE Trans. Veh. Technol. 69, 41–54. https://doi.org/10.1109/TVT.2019.2949603
Zou, Q., Ni, L., Wang, Q., Li, Q., Wang, S., 2017. Robust Gait Recognition by Integrating Inertial and RGBD Sensors. IEEE Trans. Cybern. https://doi.org/10.1109/TCYB.2017.2682280