A hybrid spatial–temporal deep learning architecture for lane detection

Dong, Yongqi; Patil, Sandeep; van Arem, Bart; Farah, Haneen

DOI
10.1111/mice.12829

Publication date
2022

Document Version
Final published version

Published in
Computer-Aided Civil and Infrastructure Engineering

DOI: 10.1111/mice.12829

RESEARCH ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

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, Delft, The Netherlands

[2] Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands

Correspondence
Haneen Farah, Department of Transport and Planning, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, The Netherlands.
Email: h.farah@tudelft.nl

Funding information
Applied and Technical Sciences (TTW), a subdomain of the Dutch Institute for Scientific Research (NWO), Grant/Award Number: 17187

Abstract

Accurate and reliable lane detection is vital for the safe performance of lane-keeping assistance and lane departure warning systems. However, under certain challenging circumstances, it is difficult to get satisfactory performance in accurately detecting the lanes from one single image as mostly done in current literature. Since lane markings are continuous lines, the lanes that are difficult to be accurately detected in the current single image can potentially be better deduced if information from previous frames is incorporated. This study proposes a novel hybrid spatial–temporal (ST) sequence-to-one deep learning architecture. This architecture makes full use of the ST information in multiple continuous image frames to detect the lane markings in the very last frame. Specifically, the hybrid model integrates the following aspects: (a) the single image feature extraction module equipped with the spatial convolutional neural network; (b) the ST feature integration module constructed by ST recurrent neural network; (c) 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 architecture can effectively handle challenging driving scenes and outperforms available state-of-the-art methods.

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 and a priority for academic researchers, manufacturers, policymakers, and their potential future users. Automated driving requires a full understanding of the environment around the automated vehicle through its sensors. Vision-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 itself within the lane. This is also the foundation of most lane-keeping assistance and lane departure warning systems (Andrade et al., 2019; Bar Hillel et al., 2014; W. Chen et al., 2020; Liang et al., 2020; Xing et al., 2018).

Traditional vision-based lane-detection methods rely on hand-crafted low-level features (e.g., color, gradient, and ridge features) and usually work in a four-step procedure, that is, image pre-processing, feature extraction, line detection and fitting, and post-processing (Bar Hillel et al., 2014; Haris & Glowacz, 2021). Traditional computer vision

techniques, for example, inverse perspective mapping (Aly, 2008; B. F. Wang et al., 2014), Hough transform (Berriel et al., 2017; Jiao et al., 2019; Zheng et al., 2018), Gaussian filters (Aly, 2008; Sivaraman & Trivedi, 2013; Y. Wang et al., 2012), and random sample consensus (Aly, 2008; Choi et al., 2018; Du et al., 2018; Guo et al., 2015; Lu et al., 2019), are usually adopted in the four-step procedure. The problems of traditional methods are: (a) hand-crafted features are cumbersome to manage and not always useful, suitable, or powerful; and (b) the detection results are always based on one single image. Thus, the detection accuracies are relatively not high.

During the last decade, with the advancements in deep learning algorithms and computational power, many deep neural network-based methods have been developed for lane detection with good performance. There are generally two dominant approaches (Tabeli et al., 2021b), that is, (1) segmentation-based pipeline (Kim & 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 a (horizontal) grid, and the model predicts the most probable location to contain a part of a lane marking in each row. Recently, Lizhe Liu et al. (2021) summarized two additional categories of deep learning-based lane-detection methods, that is, the anchor-based approach (Z. Chen et al., 2019; Li et al., 2020; Tabeli et al., 2021b; 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, which directly outputs parametric lines expressed by curve equation (R. Liu et al., 2020; Tabeli et al., 2021a). Apart from these dominant approaches, some other less common methods were proposed recently. For instance, Lin et al. (2020) fused the adaptive anchor scheme (designed by formulating a bilinear interpolation algorithm) aided informative feature extraction and object detection into a single deep convolutional neural network (CNN) for lane detection from a top-view perspective. Philion (2019) developed a novel learning-based approach with a fully convolutional model to decode the lane structures directly rather than delegating structure inference to post-processing, plus an effective approach to adapt the model to new contexts by unsupervised transfer learning.

Similar to traditional vision-based lane-detection methods, most available deep learning models utilize only the current image frame to perform the detection. Until very recently, a few studies have explored the combination of CNN and recurrent neural network (RNN) to detect lane markings or simulate autonomous driving using continuous driving scenes (Chen et al., 2020; Zhang et al., 2021; Zou et al., 2020). However, the available methods do not take full advantage of the essential properties of the lane being long continuous solid or dashed line structures. Also, they do not yet make the utmost of the spatial–temporal (ST) information together with correlation and dependencies in the continuous driving frames. Thus, for certain extremely challenging driving scenes, their detection results are still unsatisfactory.

In this paper, lane detection is treated as a segmentation task, in which a novel hybrid ST sequence-to-one deep learning architecture is developed for lane detection through a continuous sequence of images in an end-to-end approach. To cope with challenging driving situations, the hybrid model takes multiple continuous frames of an image sequence as inputs, and integrates the single image feature extraction module, the ST feature integration module, together with the encoder–decoder structure to make full use of the ST information in the image sequence. The single image feature extraction module utilizes modified common backbone networks with embedded spatial CNN (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. Next, the extracted features are fed into ST-RNN layers to capture the ST dependencies and correlations among the continuous frames. An encoder–decoder structure is adopted with the encoder consisting 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 training works in an end-to-end way as a supervised learning approach. To train and validate the proposed model on two large-scale open-sourced datasets, that is, tvtLANE (Zou et al., 2020) and TuSimple, a corresponding training strategy has been also developed. To summarize, the main contributions of this paper lie in:

1. A hybrid ST 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.
2. The proposed model architecture is the first attempt that tries to strengthen both spatial relation feature extraction in every single image frame and ST correlation together with dependencies among continuous image frames for lane detection.
3. The implementation utilized two widely used neural network backbones, that is, UNet (Ronneberger et al.,

2015) and SegNet (Badrinarayanan et al., 2017) and included extensive evaluation experiments on commonly used datasets, demonstrating the effectiveness and strength of the proposed model architecture.
4. The proposed model can tackle lane detection in challenging scenes such as curves, dirty roads, serious vehicle occlusions, and so forth, and outperforms all the available state-of-the-art baseline models in most cases with a large margin.
5. Under the proposed architecture, the light version model variant can achieve beyond state-of-the-art performance while using fewer parameters.

2 | PROPOSED METHOD

Although many sophisticated methods have been proposed for lane detection, most of the available methods use only one single image resulting in unsatisfactory performance under some extremely challenging scenarios, for example, dazzle lighting, and serious occlusion. This study proposes a novel hybrid ST sequence-to-one deep neural network architecture for lane detection. The architecture was inspired by: (a) the successful precedents of hybrid deep neural network architectures that fuse CNN and RNN to make use of information in continuous multiple frames (Zhang et al., 2021; Zou et al., 2020); (b) the domain prior knowledge that traffic lanes are long continuous shape line structure with strong spatial relationship. The architecture integrates two modules of two distinctive neural networks with complementary merits, that is, SCNN and convolutional long short-term memory (ConvLSTM) neural network, under an end-to-end encoder–decoder structure, to tackle lane detection in challenging driving scenes.

2.1 | Overview of the proposed model architecture

The proposed deep neural network architecture adopts a sequence-to-one end-to-end encoder–decoder structure as shown in Figure 1.

Here "sequence-to-one" means that the model gets a sequence of multi-images as input and outputs the detection result of the last 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, that is, 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, that is, 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. Following this, the extracted feature maps of the continuous frames, 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 obtained from the ST-RNN and decodes the content to the original input image size with the detection results. The proposed model architecture is implemented with two backbones, UNet (Ronneberger et al., 2015) and SegNet (Badrinarayanan et al., 2017). Note, in the UNet-based architecture, similar to (Ronneberger et al., 2015), the proposed model employs the skip connection between the encoder and decoder phase by concatenating operation to reuse features and retain 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 reuses 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 of 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 CNN-based encoder–decoder for image semantic-segmentation tasks in various domains (Badrinarayanan et al., 2017; S. Wang et al., 2020; Yasrab et al., 2017), this study also adopts the "symmetrical" 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

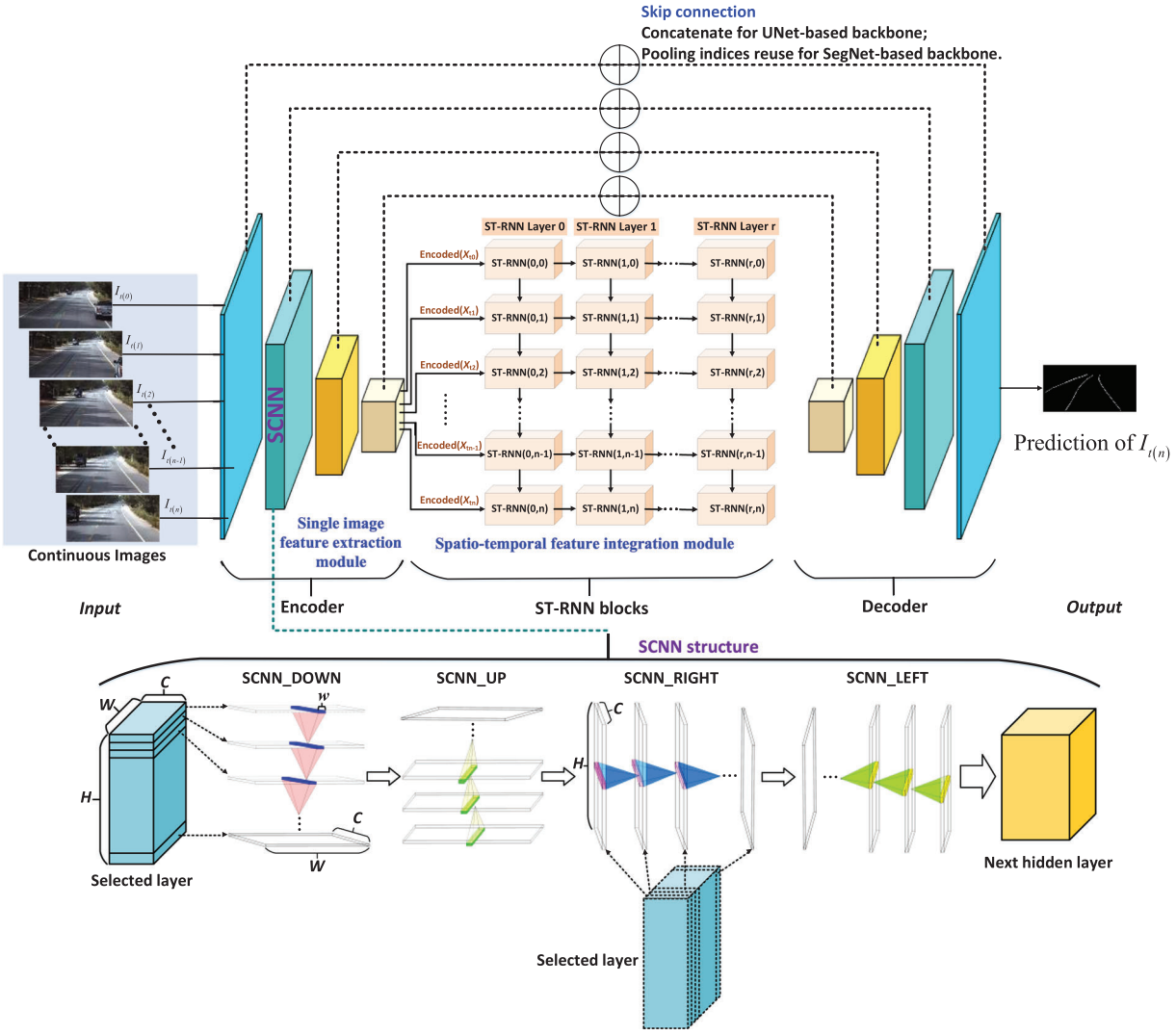


FIGURE 1 The architecture of the proposed model

as the input image size, the whole network can work in an end-to-end approach. In the implementation, two widely used backbones, UNet and Seg-Net, 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 ST correlations and dependencies among the input continuous image frames, ST-RNN blocks are embedded in the middle of the encoder–decoder networks.

2. SCNN: The 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 Figure 1.

SCNN can propagate the spatial information in one image through four directions as shown with the suffix "DOWN," "UP," "RIGHT," "LEFT" in Figure 1, which denotes downward, upward, rightward, and leftward, respectively. Take the "SCNN_DOWN" module for an example, considering that SCNN is adopted on a three-dimensional tensor of size $C \times W \times H$, wherein the lane-detection task, C, W, and H denote the number of channels, image (or its feature map) width, and heights, respectively. For SCNN_D, the input tensor would be split into H slices, and the first slice will then be sent into a convolution operation layer with C kernels of size $C \times w$, in which w is the kernel width. Different from the traditional CNN in which the output of one convolution layer is introduced into the next layer directly, 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 in the selected direction is updated. The convolution kernel weights are shared throughout 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, for example, 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 ST correlations in the continuous frames is proposed in this paper.

3. ST-RNN module: In this proposed framework, the multiple continuous frames of images are modeled as "image-time-series" inputs. To capture the ST 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 ST information to the decoder.

Various versions of RNNs have been proposed, for example, LSTM together with its multivariate version, that is, fully connected LSTM (FC-LSTM), and gated recurrent unit (GRU), to tackle time-series data in different application domains. In this paper, two state-of-the-art RNN networks, that is, ConvLSTM (Shi et al., 2015) and convolutional GRU (ConvGRU; Ballas et al., 2016), are employed. These models, considering their abilities in ST feature extraction, generally outperform other traditional RNN models.

A general critical problem for the vanilla RNN model is the gradients vanishing (Hochreiter & Schmidhuber, 1997; Pascanu et al., 2013; Ribeiro et al., 2020). For this, LSTM introduces memory cells and gates to control the information flow to trap the gradient preventing it from vanishing during the backpropagation. 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, the ConvLSTM (Shi et al., 2015) is selected to build the ST-RNN block of the proposed framework. 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 Equations (1)–(5), where $\odot$ denotes the Hadamard product, $*$ denotes the convolution operation, $\sigma(\cdot)$ represents the sigmoid function, and $\tanh(\cdot)$ represents the hyperbolic tangent function; X_t, $\mathcal{C}_t$, and $\mathcal{H}_t$ are the input (i.e., the extracted features from the encoder in the proposed 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 that 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, for example, b_i is the input gate's bias.

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

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

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

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

$$\mathcal{H}_t = o_t \odot \tanh\left(\mathcal{C}_t\right) \quad (5)$$

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 by Equations (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 $\tilde{\mathcal{H}}$ represents the current candidate hidden representation.

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

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

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

$$\mathcal{H}_t = z_t \tilde{\mathcal{H}} + \left(1 - z_t\right) \mathcal{H}_{t-1} \quad (9)$$

In ConvGRU, there are only two gate structures, that is, the updated gate z_t and the reset gate r_t. It is the update gate z_t that decides how to update the hidden representation when generating the ultimate result of $\mathcal{H}_t$ at the current layer as shown in Equation (9), while the reset gate r_t is served to control to what extent the feature information captured in the previous hidden state is supposed to be forgotten through an element-wise multiplication operation when calculating current candidate hidden representation. From the equations, it is concluded that the

information of $\mathcal{H}_t$ mainly comes from $\tilde{\mathcal{H}}_t$, while $\mathcal{H}_{t-1}$ as the previous hidden-state representation also contributes to the process of computing the final representation of $\mathcal{H}_t$; thus, the temporal dependencies are captured.

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

2.3 | Detailed implementation

1. Network design details: The proposed ST sequence-to-one neural network was developed for the lane-detection task with K (in this paper $K = 5$ if not specified) continuous image frames as inputs. The image frames were first fed into the encoder for feature extraction and abstraction. Different from the normal CNN-based encoder, the SCNN layer was utilized to effectively extract the spatial relationships within every image. Different locations of the SCNN layer were tested, that is, embedding the SCNN layer after the first hidden convolutional layer or at the very beginning. The outputs of the encoder network were 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 ST dependencies and correlations among different image frames. In short, the encoder network is primarily responsible for spatial feature extraction and abstraction transforming input images into 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 ST dependencies.

The outputs of the ST-RNN blocks were 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 to reuse the retained features from previous encoder layers for more accurate predictions at the decoder phase. After the decoder phase, the lane-detection result is obtained as an image in the equivalent size to the 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 Tables A1 and A2.

For both SegNet-based and UNet-based implementations, two types of RNN layers, that is, ConvLSTM and ConvGRU, were tested to serve as the ST-RNN block. Besides, the ST-RNN blocks were tested with one and two hidden layers. So there are four variants of in the proposed SegNet-based models, that is, 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 SCNN layer-embedded encoder and one hidden layer of ConvGRU as the ST-RNN block. This naming rule applies to the other three variants as well. Also, there are four variants of the proposed UNet-based models, with a similar naming rule.

In the proposed models with UNet as the backbone, the number of kernels used in the last convolutional block of the encoder part differs from the original UNet'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. This is done, similar to (Zou et al., 2020), to better connect the output of the encoder with the ST-RNN block (ConvLSTM or ConvGRU layers). To do so, the parameters of the full-connection layer are designed to be quadrupled, while the side lengths of the feature maps are reduced to half, at the same time, the number of kernels remains unchanged. This strategy also somewhat contributes to reducing the parameter size of the whole network.

A modified light version of UNet (UNetLight) was also tested to serve 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 three unchanged) and *Output* in *Out_ConvBlock* (with the output channel of two unchanged). To save space, the parameter settings of UNetLight-based implementation will not be illustrated.

2. Loss function: Since the lane detection is modeled as a segmentation task and a pixel-wise binary classification problem, cross-entropy is a suitable candidate to serve as the loss function. However, because the

pixels classified to be lanes are always quite less than those classified to be the background (meaning that it is an imbalanced binary classification and discriminative segmentation task), in the implementation, the loss was built upon the weighted cross-entropy. The adopted loss function as the standard weighted binary cross-entropy function is given as in Equation (10),

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

where S is number of training examples, w stands for the weight, which is set according to the ratio between the total lane pixel quantities and non-lane pixel quantities 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: The proposed neural networks with different variants, together with the baseline models were trained on the Dutch high-performance supercomputer clusters, Cartesius and Lisa, using 4 Titan RTX GPUs with the data parallel mechanism in PyTorch. The input image size was set as 128×256 to reduce the computational payload. The batch size was set to be as large as possible (e.g., 64 for UNet-based network architecture, 100 for SegNet-based ones, and 136 for UNetLight-based ones), and the learning rate was initially set to 0.03. The Rectified Adam (RAdam) optimizer (Liyuan Liu et al., 2019) was first used in this work for training the model at the beginning. At the later stage, when the training accuracy was beyond 95%, the optimizer was switched to the stochastic gradient descent (Bottou, 2010) optimizer with decay. With the labeled ground truth, the models were trained through iteratively updating the parameters in the weight matrixes and the losses on the basis of the deviation between outputs of the proposed neural network and the ground truth using the backpropagation mechanism. To speed up the training process, the pre-trained weights of SegNet and UNet on ImageNet (Deng et al., 2009) were adopted.

3 | EXPERIMENTS AND RESULTS

Extensive experiments were carried out to inspect and verify the accuracy, effectiveness, and robustness of the proposed lane-detection model using two large-scale open-sourced datasets. The proposed models were evaluated on different driving scenes and were compared with several state-of-the-art baseline lane-detection methods, which also employ deep learning, for example, UNet (Ronneberger et al., 2015), Seg-Net (Badrinarayanan et al., 2017), 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

1. tvtLANE training set: To verify the proposed model performance, the tvtLANE dataset (Zou et al., 2020) based upon the TuSimple lane marking challenge dataset, was first utilized for training, validating, and testing. The original dataset of the TuSimple lane marking challenge includes 3626 clips of training and 2782 clips of testing, which are collected under various weather conditions and during different periods. In each clip, there are 20 continuous frames saved in the same folder. In each clip, only the lane marking lines of the very last frame, that is, the 20th frame, are labeled with the ground truth officially. Zou et al. (2020) additionally labeled every 13th image in each clip and added their own collected lane dataset, which includes 1148 sequences of rural driving scenes collected in China. This immensely expanded the variety of the road and driving conditions since the original TuSimple dataset only covers the highway driving conditions. K continuous frames 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, crop, flip, and rotation operations were employed; thus, a total number of $(3626 + 1148) \times 4 = 19{,}096$ continuous sequences were produced, in which 38,192 images are labeled with ground truth. To adapt to different driving speeds, the input image sequences were sampled at three strides with a frame interval of one, two, or three. Then, three sampling methods were employed to construct the training samples regarding the labeled 13th and 20th frames in each sequence as demonstrated in Table 1.

2. tvtLANE testing set: Two different datasets were used for testing, that is, testset #1 (normal) and testset #2 (challenging), which are also formatted with five continuous images as the input to detect the lane markings in the very last frame with the labeled ground truth. To be specific, testset #1 is built upon the original TuSimple test set for normal driving scene testing; while testset #2 is constructed with 12 challenging driving situations, especially used for robustness evaluation. The detailed descriptions of the trainset and testset in tvtLANE are illustrated in Table 1, with examples shown in Figure 2.

TABLE 1 Trainset and testset in tvtLANE

Trainset				
Subset				**Labeled images num**
Original TuSimple dataset (Highway)				7252
Zou et al. (2020) added (rural road)				2296
Sample methods				
Labeled ground truth	**Sample stride**			**Train sample frames**
13th	3			First, fourth, seventh, 10th, 13th
	2			Fifth, seventh, nine, 11th, 13th
	1			Nine, 10th, 11th, 12th, 13th
20th	3			Eight, 11th, 14th, 17th, 20th
	2			12th, 14th, 16th, 18th,20th
	1			16th, 17th, 18th, 19th, 20th
Testset				
Subset	**Labled images num**	**Labled ground truth**	**Sample stride**	**Test sample frames**
Testset #1 normal	540	13th	1	Ninth, 10th, 11th, 12th, 13th
		20th	1	16th, 17th, 18th, 19th, 20th
Testset #2 challenging	728	All	1	First, second, third, fourth, fifth
				Second, third, fourth, fifth, sixth
				Third, fourth, fifth, sixth, seventh
				…

FIGURE 2 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

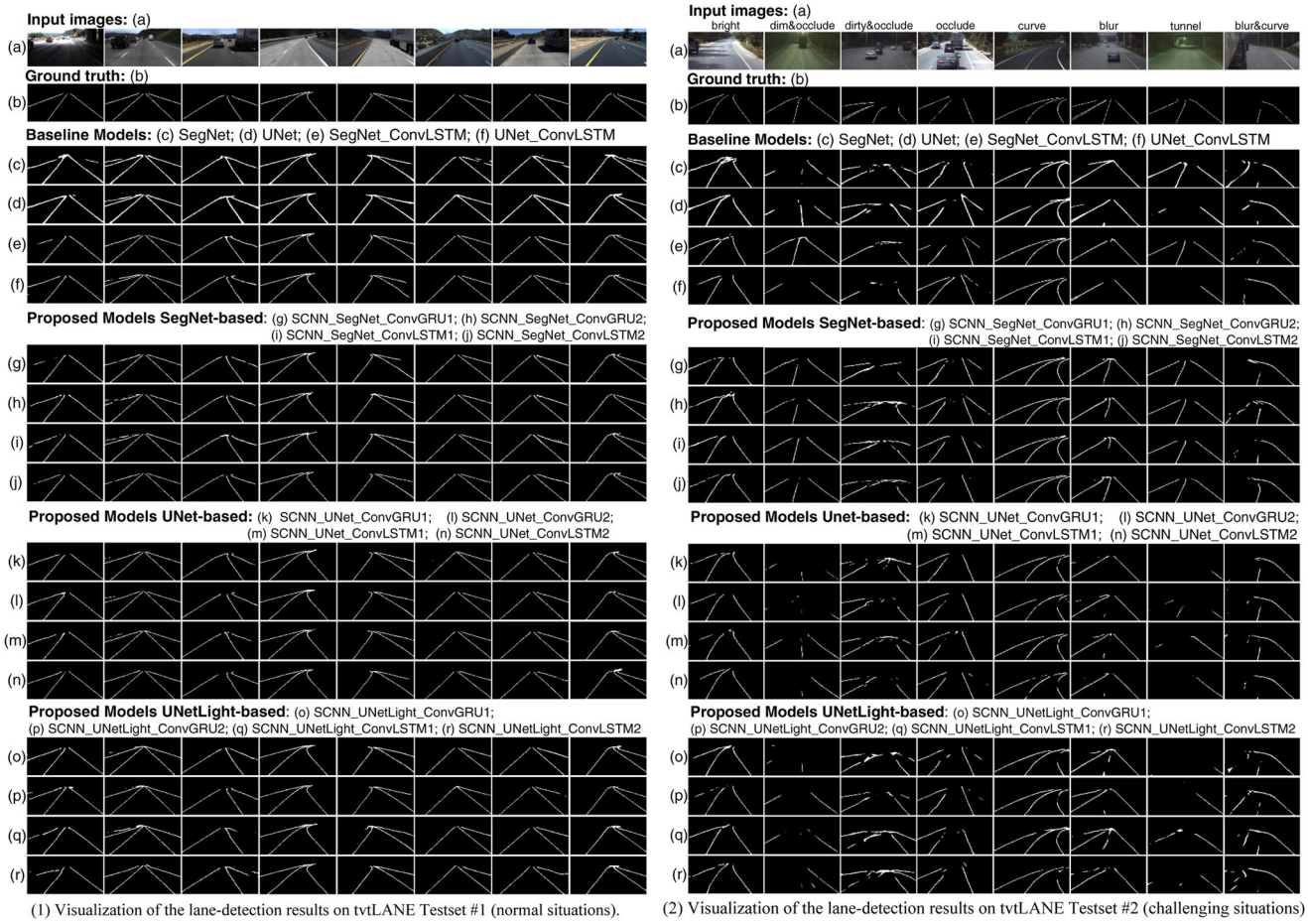
Qualitative evaluation with the visualization of the lane detection results is the most intuitive approach to compare and evaluate the properties of different models, and it helps to find insights regarding their pros and cons.

1. tvtLANE testset #1: Normal situations

Samples of the lane-detection results on tvtLANE testset #1 of the proposed models and other state-of-the-art models are demonstrated in Figure 3(1). All these results are without post-processing.

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

1. The number of lines needs to be predicted correctly. A wrong detection or a misprediction might cause the automated vehicles to consider unsafe or unreachable areas as drivable areas resulting in potential accidents. As illustrated in the first and second columns in Figure 3(1), the proposed models can identify the correct number of lane lines, while the baseline models, especially the ones using a single image, somewhat cannot detect the correct number of lines, compared with ground truth.
2. The positions of each lane marking line should be predicted precisely accords with the ground truth. As illustrated in Figure 3(1), the proposed models in row (j) with the model named by SCNN_SegNet_ConvLSTM2 and row (n) with the model named SCNN_UNet_ConvLSTM2, could deliver better lane location predictions with thinner lines, compared with the baseline models. Superior to scattering points around, thinner predicted lane lines indicate a more precise model prediction of the lane position.
3. The predicted lane lines should not merge or be broken. As illustrated in the first, second, sixth, seventh, and eighth columns of Figure 3(1), some baseline models' output lane lines either merge at the far end or break the continuity with dashed lines. The proposed models perform slightly better although in a few cases the lines are also discontinuous.



(1) Visualization of the lane-detection results on tvtLANE Testset #1 (normal situations).

(2) Visualization of the lane-detection results on tvtLANE Testset #2 (challenging situations)

FIGURE 3 Qualitative evaluation: Visualization of the lane-detection results on (1) tvtLANE testset #1 and (2) tvtLANE testset #2

4. The lanes should be predicted correctly even at the boundary of the image. As can be found in Figure 3(1), some baseline models, for example, rows (c), (d), and (e), run across difficulties at the top boundary of the image with merge lanes on the top. This also accords with the aforementioned property.
5. The lane-detection models should deliver accurate predictions under different driving scenes, even under some challenging situations. For example, in the second, third, fifth, and seventh columns of Figure 3(1), vehicles are 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

Figure 3(2) shows the comparison of the proposed models with the baseline models under some extremely challenging driving scenes in the tvtLANE testset #2. All the results are not post-processed. 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 extremely challenging cases, the lanes are totally occluded by vehicles, other objects, and/or shadows, which could be very difficult even for humans to do the detection.

As can be observed in Figure 3(2), although all the baseline models fail in these challenging cases, the proposed models, especially the one named SCNN_SegNet_ConvLSTM2 illustrated in the row (k), could still deliver good predictions in almost every situation listed in Figure 3(2). The only flaw is that in the third 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 fourth, seventh, and eighth columns, the robustness of SCNN_SegNet_ConvLSTM2's property in detecting the correct number of lane lines is further verified, especially, one can observe in the fourth column, where almost all the other models are defeated, SCNN_SegNet_ConvLSTM2 can still predict the correct number of lanes.

Furthermore, it should be noticed that 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 third to fifth rows.

3.3 | Quantitative evaluation

1. Evaluation metrics: This subsection examines the proposed models' properties regarding quantitative evaluations. When treated as a pixel-wise classification task, accuracy must be the most simple criterion for the performance evaluation of lane detection (Zou et al., 2017), which represents the overall classification performance in terms of correctly classified pixels, indicated in Equation (11).

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

However, since it is an imbalanced binary classification problem, where the lanes pixels are far less than the background pixels, using only accuracy to evaluate the model is not suitable. Thus, precision, recall, and F-measure, illustrated by Equations (12)–(14), are commonly employed.

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

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

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

In the above equations, true positive indicates the number of image pixels that are lane marking and are correctly identified; false positive means the number of image pixels that are background but are wrongly classified as lane markings; false negative stands for the number of image pixels that are lane marking but are wrongly classified as the background.

Specifically, this study chooses $\beta = 1$, which corresponds to the F1-measure (harmonic mean) shown in Equation (15).

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

The F1-measure, which balances precision and recall, is always selected as the main benchmark for model evaluation (e.g., Lizhe Liu et al., 2021; Pan et al., 2018; Xu et al., 2020; Zhang et al., 2021; Zou et al., 2020).

Furthermore, the model parameter size, that is, Params (M), together with the multiply-accumulate (MAC) operations, that is, MACs (G), are provided as indicators of the model complexity. The two indicators are commonly used to estimate models' computational complexities and real-time capabilities.

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 the quantitative evaluation with the qualitative evaluation, it could be easily interpreted that the highest precision, accuracy, and F1-measure are mainly derived from (i) the correct lane number, (ii) the accurate lane position, (iii) the sound continuity in the detected lanes, and (iv) the thinness of the predicted lanes with less blurriness, which accords with (ii). The correct prediction directly reduces the number of false positives, and a good precision contributes to better accuracy and F1-measure. Considering the structure of the proposed model architecture, a further explanation of the high F1-measure, accuracy, and precision can be explained as follows:

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

Second, the ST-RNN blocks, that is, 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 the best of the ST information among the processed K continuous frames by regulating the weights of the convolutional kernels within the SCNN and ConvLSTM/ConvGRU layers.

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

Looking at the main metric, F1-measure, it is demonstrated that increasing only precision or only recall will not improve the F1-measure. Although the bassline models of UNet, SegNet, and SegNet_ConvLSTM get better recalls, they do not deliver good F1-measure since their precisions are much lower than the proposed model of SCNN_SegNet_ConvLSTM2 or

TABLE 2 Model performance comparison on tvtLANE testset #1 (normal situations)

		Test_Acc (%)	Precision	Recall	F1-measure	MACs (G)	Params (M)
Models using a single image as input	**Baseline models**						
	UNet	96.54	0.790	**0.985**	0.877	15.5	13.4
	SegNet	96.93	0.796	0.962	0.871	50.2	29.4
	SCNN*	96.79	0.654	0.808	0.722	77.7	19.2
	LaneNet*	97.94	0.875	0.927	0.901	44.5	19.7
Models using continuous images sequence as inputs	SegNet_ConvLSTM**	97.92	0.874	0.931	0.901	217.0	67.2
	UNet_ConvLSTM**	98.00	0.857	0.958	0.904	69.0	51.1
	Proposed models (SegNet-based)						
	SCNN_SegNet_ConvGRU1	98.00	0.878	0.935	0.905	219.2	43.7
	SCNN_SegNet_ConvGRU2	98.05	0.888	0.918	0.903	221.5	57.9
	SCNN_SegNet_ConvLSTM1	98.01	0.881	0.935	0.907	220.0	48.5
	SCNN_SegNet_ConvLSTM2	98.07	**0.893**	0.928	0.910	223.0	67.3
	Proposed models (UNet-based)						
	SCNN_UNet_ConvGRU1	98.13	0.878	0.957	0.916	77.9	27.7
	SCNN_UNet_ConvGRU2	**98.19**	0.887	0.950	0.917	87.0	41.9
	SCNN_UNet_ConvLSTM1	98.18	0.886	0.948	0.916	81.0	32.4
	SCNN_UNet_ConvLSTM2	**98.19**	0.889	0.950	**0.918**	93.0	51.3
	Proposed models (light version UNet-based)						
	SCNN_UNetLight_ConvGRU1	97.83	0.850	0.960	0.902	19.6	**6.9**
	SCNN_UNetLight_ConvGRU2	98.01	0.863	0.950	0.905	21.9	10.5
	SCNN_UNetLight_ConvLSTM1	97.71	0.830	0.950	0.886	20.4	8.1
	SCNN_UNetLight_ConvLSTM2	97.76	0.840	0.953	0.893	23.4	12.8

Abbreviations: ConvGRU, convolutional gated recurrent unit; ConvLSTM, convolutional long short-term memory; MAC, multiply-accumulate; SCNN, spatial convolutional neural network; UNetLight, modified light version of UNet.

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

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

SCNN_UNet_ConvLSTM2. Regarding the good recall of UNet and SegNet, it could be speculated from the qualitative evaluation, where one can find that UNet 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, the models were evaluated on a brand-new dataset, that is, the tvtLANE testset #2. As introduced in Section 3.1, tvtLANE testset #2 includes 728 images in highway, urban, and rural driving scenes. These challenging driving scenes' data were obtained by data recorders at various locations, outside and inside the car front windshield under different road and weather conditions. Testset #2 is a challenging and comprehensive dataset for model evaluation, from which some cases would be difficult enough for humans to do the correct detection.

Table 3 demonstrates 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, Table 3 provides the precision and F1-measure for the evaluation reference.

As indicated by the bold numbers, the proposed model, SCNN_SegNet_ConvLSTM2, results in the best

TABLE 3 Model performance comparison on tvtLANE testset #2 (12 types of challenging scenes)

Models/challenging Scenes	*1-curve &occlude*	*2-shadow-bright*	*3-bright*	*4-occlude*	*5-curve*	*6-dirty &occlude*	*7-urban*	*8-blur &curve*	*9-blur*	*10-shadow-dark*	*11-tunnel*	*12-dim &occlude*	*Overall*
Precision													
UNet	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													
UNet	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

Abbreviations: ConvGRU, convolutional gated recurrent unit; ConvLSTM, convolutional long short-term memory; SCNN, spatial convolutional neural network; UNetLight, modified light version of UNet.

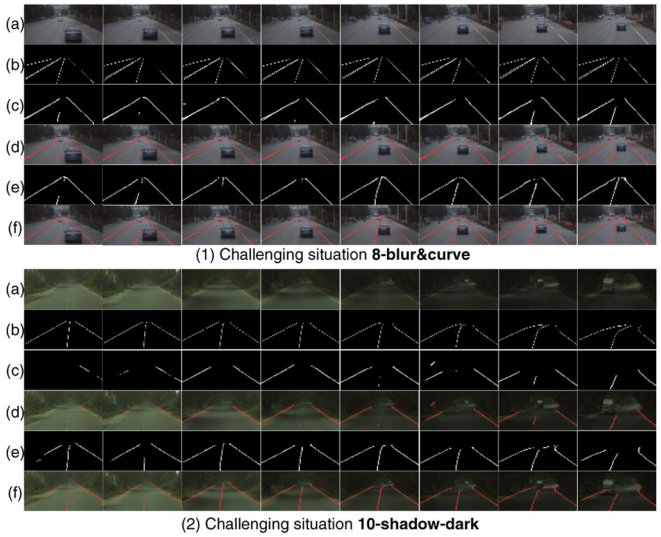


FIGURE 4 Visual comparison of the lane-detection results on challenging driving situations for UNet_ConvLSTM and the proposed model SCNN_SegNet_ConvLSTM2. All the results are not post-processed. (a) Input images. (b) Ground truth. (c) Detection results of UNet_ConvLSTM. (d) Detection results of UNet_ConvLSTM overlapping on the original images. (e) Detection results of SCNN_SegNet_ConvLSTM2. (f) 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-dark*

F1-measure at the overall level and in more situations, while the UNet_ConvLSTM results in the best precision at the overall level and in more situations. Incorporating with the qualitative evaluation in Figure 3(2), it is shown that UNet_ConvLSTM tends to not classify pixels into lane lines for uncertain areas under some challenging situations (e.g., the second and seventh columns in Figure 3(2)). This might be the reason for its obtaining better precision. To further confirm this speculation, Figure 4 compares the lane-detection results of SCNN_SegNet_ConvLSTM2 and UNet_ConvLSTM under challenging situations *8-blur&curve*, and *10-shadow-dark*, where UNet_ConvLSTM delivers very good precisions.

As illustrated in Figure 4, 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, the 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, it can be concluded that the 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 almost all 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, the effects of SCNN were investigated by evaluating the performances of the model variants with and without SCNN layers. As demonstrated in Figures 3 and 4, together with the quantitative results in Tables 2 and 3, the proposed SegNet and UNet-based models with SCNN-embedded encoder, that is, 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 but without SCNN. 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, as shown in Table 2, 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 the 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.

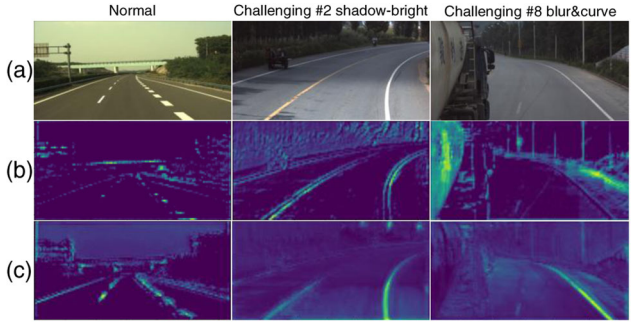


FIGURE 5 Visualization of the extracted low-level features at Down_ConvBlock_1 for UNet-based models. (a) Original image. (b) Results of UNet_ConvLSTM (without SCNN layers). (c) Results of the SCNN_UNet_ConvLSTM2 (with SCNN layers)

Figure 5 visualizes the extracted 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 Figure 5b, 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, that is, there are some interference signals, especially as visualized in the third image of row (b), 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 the marking detection results. It might further influence the final detection results. In contrast, with SCNN layers, the extracted features of the lanes are more inerratic, clear, and evident as shown in Figure 5c. 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 that the adding of the SCNN layer embedded in the encoder does contribute to the spatial feature extraction, with which the model could better make the utmost use of the ST information among the continuous image frames.

2. Different locations of SCNN layer

Results of testing different locations of the SCNN layer in the proposed model architecture 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.

3. Type and number of ST-RNN layers

As described in Section 3, in the proposed model architecture two types of RNNs, That Is, ConvLSTM and ConvGRU, are employed to serve in the ST-RNN block, to capture and make use of the ST dependencies and correlations among the continuous image sequences. The number of hidden ConvLSTM and ConvGRU layers were also tested from 1 to 2. The quantitative results are demonstrated in Tables 2 and 3, while some intuitive qualitative insights could be drawn from Figures 3 and 4.

From Table 2, it is illustrated that in general models adopting ConvLSTM layers in the ST-RNN block perform better than those adopting ConvGRU layers with improved F1-measure, except for the UNetLight-based models. This could be explained by ConvLSTM's better properties in extracting ST features and capturing time dependencies by more control gates and thus more parameters, compared with ConvGRU. Furthermore, from Tables 2 and 3, it is observed that models with two hidden ST-RNN layers, for both ConvLSTM and ConvGRU, generally perform better than those with only one hidden ST-RNN layer. This could be speculated that with two hidden ST-RNN layers, one layer can serve for sequential feature extraction, and the other can achieve ST feature integration. The improvements of two ST-RNN layers over one are not that significant, which might be due to (a) models employing one ST-RNN layer already obtaining good results; (b) since the length of the continuous image frames is only five, one ST-RNN layer might be already enough to do the ST feature extraction, so when incorporating longer image sequences, the superiorities of two ST-RNN layers could be promoted. However, longer image sequences require more computational resources and longer training time, which could not be afforded at the present stage in this study. 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, that is, SCNN_SegNet_ConvLSTM2 and SCNN_UNet_ConvLSTM2, possess a bit more parameters, compared with the baseline SegNet_ConvLSTM and

TABLE 4 Model performance comparison with different locations of SCNN layer on tvtLANE testset #1 and #2

Models/testing datasets	Location of SCNN	Testset #1 (normal situations) Test_Acc (%)	Precision	Recall	F1-Measure	Testset #2 (challenging scenes) 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_Conv LSTM	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

Abbreviations: ConvLSTM, convolutional long short-term memory; SCNN, spatial convolutional neural network.

UNet_ConvLSTM, respectively. However, almost all of the proposed model variants with different types and numbers of ST-RNN layers outperform the baselines, and some of them are even with low parameter sizes, for example, 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, four model variants were implemented with a modified light version of UNet, that is, UNetLight, serving as the network backbone to reduce the total parameter size and improve the model's ability to operate in real-time. 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 three unchanged) and Output in *Out_ConvBlock* (with the output channel of two unchanged). From the testing results in Table 2, it is shown 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, that is, UNet_ConvLSTM, SCNN_UNetLight_ConvGRU2 only uses less than one-fifth of the parameter size but delivers better evaluation metrics in testing accuracy, precision, and F1-measure.

Regarding UNetLight-based models, models using ConvGRU layers in the ST-RNN block perform better than those adopting ConvLSTM. The reason could be that light version UNet cannot 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 ST sequence-to-one model framework with a hybrid neural network architecture is proposed for robust lane detection under various normal and challenging driving scenes. This architecture integrates a single image feature extraction module with SCNN, ST feature integration module with ST-RNN, together with the encoder–decoder structure. The proposed architecture achieved significantly better results in comparison to baseline models that use a single frame (e.g., UNet, SegNet, and LaneNet), as well as the state-of-art models adopting "CNN+RNN" structures (e.g., UNet_ConvLSTM, SegNet_ConvLSTM), with the best testing accuracy, precision, F1-measure on the normal driving dataset (i.e., tvtLANE testset #1) and the best F1-measure on 12 challenging driving scenarios dataset (tvtLANE testset #2). The results demonstrate the effectiveness of strengthening spatial relation abstraction in every single image with SCNN layer, plus the employment of multiple continuous image sequences as inputs. The results also demonstrate the proposed model architecture's ability in making the best of the ST information in continuous image frames. Extensive experimental results show the superiorities of the sequence-to-one "SCNN + ConvLSTM" over "SCNN + ConvGRU" and ordinary "CNN + ConvLSTM" regarding sequential ST feature extracting and learning, together with target-information classification for robust lane detection. In addition, 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 ST correlations and dependencies among image frames for lane detection, and the extensive evaluation experiments demonstrate the strength of this proposed architecture. Therefore, it is recommended

in future research to incorporate both aspects to obtain better performance.

In this paper, the challenging cases do not include night driving, rainy, or wet road conditions, nor do they include situations in which the input images are defective (e.g., partly masked or blurred). There are demands to build larger test sets with comprehensive challenging situations to further validate the model's robustness. Since a large amount of unlabeled driving scene data involving various challenging cases was collected within the research group, 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. Furthermore, to further enhance the lane-detection model, customed loss function, pre-trained techniques adopted in image-inpainting task, for example, masked autoencoders, plus sequential attention mechanism could be introduced and integrated into the proposed framework.

ACKNOWLEDGMENTS

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.

REFERENCES

Aly, M. (2008). Real time detection of lane markers in urban streets. *2008 IEEE Intelligent Vehicles Symposium*. Eindhoven, the Netherlands (pp. 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 Transactions on Intelligent Transportation Systems*, *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 Transactions on Pattern Analysis and Machine Intelligence*, *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. *4th International Conference on Learning Representations, ICLR 2016–Conference Track Proceedings*, San Juan, Puerto Rico.

Bar Hillel, A., Lerner, R., Levi, D., & Raz, G. (2014). Recent progress in road and lane detection: A survey. *Machine Vision and Applications*, *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 and Vision Computing*, *68*, 64–75. https://doi.org/10.1016/j.imavis.2017.07.005

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT 2010–19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, Paris, France. https://doi.org/10.1007/978-3-7908-2604-3_16

Chen, S., Leng, Y., & Labi, S. (2020). A deep learning algorithm for simulating autonomous driving considering prior knowledge and temporal information. *Computer-Aided Civil and Infrastructure Engineering*, *35*, 305–321. https://doi.org/10.1111/mice.12495

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. *Journal of Traffic and Transportation Engineering (English Edition)*, *7*(6), 748–774. 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 Intelligent Vehicles Symposium 2019*, Paris, France (pp. 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. *International Journal of Computer and Information Engineering*, *12*(4), 245–248.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on computer vision and Pattern Recognition*, Miami, FL (pp. 248–255). https://doi.org/10.1109/cvprw.2009.5206848

Du, H., Xu, Z., & Ding, Y. (2018). The fast lane detection of road using RANSAC algorithm. In J. Abawajy, K. K. Choo, & R. Islam (Eds.), *Advances in Intelligent Systems and Computing* (pp. 1–7). Springer. 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. *IEEE 12th International Symposium on Autonomous Decentralized Systems, ISADS 2015*, Taichung, Taiwan (pp. 285–288). https://doi.org/10.1109/ISADS.2015.24

Haris, M., & Glowacz, A. (2021). Lane line detection based on object feature distillation. *Electron*, *10*(9), 1102. https://doi.org/10.3390/electronics10091102

Hochreiter, S., & Schmidhuber, J. (1997). Long short term memory. *Neural Computation. Neural Computation*, *9*(8), 1735–1780.

Hou, Y., Ma, Z., Liu, C., Hui, T. W., & Loy, C. C. (2020). Inter-region affinity distillation for road marking segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA (pp. 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. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *233*(9), 2301–2311. 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. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, HI, pp. 1194–1202. 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. *2011 IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany (pp. 163–168). 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 Transactions on Intelligent Transportation Systems*, *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. *Journal of Computer Science and Technology*, *35*, 493–505. https://doi.org/10.1007/s11390-020-0476-4

Lin, C., Li, L., Cai, Z., Wang, K. C. P., Xiao, D., Luo, W., & Guo, J. G. (2020). Deep learning-based lane marking detection using A2-LMDet. *Transportation Research Record*, *2674*(11), 625–635. https://doi.org/10.1177/0361198120948508

Liu, L., Chen, X., Zhu, S. & Tan, P. (2021). CondLaneNet: A top-to-down lane detection framework based on conditional convolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 3773–3782).

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. In International Conference on Learning Representations.

Liu, R., Yuan, Z., Liu, T., & Xiong, Z. (2020). End-to-end lane shape prediction with transformers. *2021 IEEE Winter Conference on Applications of Computer Vision*, Waikoloa, HI (pp. 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. *2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV (pp. 1394–1399) https://doi.org/10.1109/IV47402.2020.9304613

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)*, *19*(18), 4028. https://doi.org/10.3390/s19184028

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M. & Van Gool, L. (2017). Fast scene understanding for autonomous driving. arXiv preprint arXiv:1708.02550.

Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., & Van Gool, L. (2018). Towards end-to-end lane detection: An Instance segmentation approach. *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China (pp. 286–291). https://doi.org/10.1109/IVS.2018.8500547

Pan, X., Shi, J., Luo, P., Wang, X., & Tang, X. (2018). Spatial as deep: Spatial CNN for traffic scene understanding. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. New Orleans, LA (pp. 7276–7283).

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *30th International Conference on Machine Learning, ICML 2013*, Atlanta, GA.

Philion, J. (2019). FastDraw: Addressing the long tail of lane detection by adapting a sequential prediction network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA (pp. 11574–11583). https://doi.org/10.1109/CVPR.2019.01185

Qin, Z., Wang, H., & Li, X. (2020). Ultra fast structure-aware deep lane detection. In A. Vedaldi, H. Bischof, T. Brox, & J. M. Frahm (Eds.), *Computer Vision–ECCV 2020: 16th European Conference* (pp. 276–291). Springer International Publishing.

Ribeiro, A. H., Tiels, K., Aguirre, L. A., & Schön, T. (2020). Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness. *International Conference on Artificial Intelligence and Statistics*, Palermo, Sicily, Italy (pp. 2370–2380).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. Wells, & A. Frangi A. (Eds.), *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015* (Vol. 9351, pp. 234–241). https://doi.org/10.1007/978-3-319-24574-4_28

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. *Advances in Neural Information Processing Systems*, Montreal, Canada.

Sivaraman, S., & Trivedi, M. M. (2013). Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. *IEEE Transactions on Intelligent Transportation Systems*, *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. *Advances in Neural Information Processing Systems*, Montreal, Canada (pp. 3104–3112).

Tabelini, L., Berriel, R., Paixão, T. M., Badue, C., de Souza, A. F. & Oliveira-Santos, T. (2021a). "PolyLaneNet: Lane Estimation via Deep Polynomial Regression," *2020 25th International Conference on Pattern Recognition (ICPR)*, (2021, pp. 6150–6156). https://doi.org/10.1109/ICPR48806.2021.9412265.

Tabelini, L., Berriel, R., Paixão, T. M., Badue, C., De Souza, A. F. & Olivera-Santos, T. (2021b). Keep your eyes on the lane: Real-time attention-guided lane detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 294–302).

Wang, B. F., Qi, Z. Q., & Ma, G. C. (2014). Robust lane recognition for structured road based on monocular vision. *Journal of Beijing Institute of Technology (English Edition)*, *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*, *8*, 7313–7322. 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*, *92*(2), 319–334. 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 Journal of Selected Topics in Signal Processing*, *11*(8), 1289–1300. https://doi.org/10.1109/JSTSP.2017.2756439

Xing, Y., Lv, C., Chen, L., Wang, H., Wang, H., 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 Journal of Automatica Sinica*, *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. In: Vedaldi A., Bischof H., Brox T., Frahm JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, (vol. 12360). Springer, Cham. https://doi.org/10.1007/978-3-030-58555-6_41

Yasrab, R., Gu, N., & Zhang, X. (2017). An encoder-decoder based Convolution Neural Network (CNN) for future Advanced Driver Assistance System (ADAS). *Applied Science*, *7*(4), 312. https://doi.org/10.3390/app7040312

Yoo, S., Lee, H. S., Myeong, H., Yun, S., Park, H., Cho, J., & Kim, D. H. (2020). End-to-end lane marker detection via row-wise classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Seattle, WA (pp. 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 Transactions on Intelligent Transportation Systems*, 1–13. 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 Recognition and Image Analysis*, *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 Transactions on Vehicular Technology*, *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 Transactions on Cybernetics*, *48*(4), 1136–1150. https://doi.org/10.1109/TCYB.2017.2682280

How to cite this article: Dong, Y., Patil, S., van Arem, B., & Farah, H. (2022). A hybrid spatial–temporal deep learning architecture for lane detection. *Computer-Aided Civil and Infrastructure Engineering*, 1–20. https://doi.org/10.1111/mice.12829

APPENDIX

TABLE A1 Parameter settings for each layer of the SegNet-based neural network

Layer		Input (channel × hight × width)	Output (channel × hight × width)	Kernel	Padding	Stride	Activation
Down_ConvBlock_1	Conv_1_1	3 × 128 × 256	64 × 128 × 256	3 × 3	(1,1)	1	ReLU
	Conv_1_2	64 × 128 × 256	64 × 128 × 256	3 × 3	(1,1)	1	ReLU
	Maxpool1	64 × 128 × 256	64 × 64 × 128	2 × 2	(0,0)	2	—
SCNN	SCNN_Down	64 × 1 × 128	64 × 1 × 128	1 × 9	(0,4)	1	ReLU
	SCNN_Up	64 × 1 × 128	64 × 1 × 128	1 × 9	(0,4)	1	ReLU
	SCNN_Right	64 × 64 × 1	64 × 64 × 1	9 × 1	(4,0)	1	ReLU
	SCNN_Left	64 × 64 × 1	64 × 64 × 1	9 × 1	(4,0)	1	ReLU
Down_ConvBlock_2	Conv_2_1	64 × 64 × 128	128 × 64 × 128	3 × 3	(1,1)	1	ReLU
	Conv_2_2	128 × 64 × 128	128 × 64 × 128	3 × 3	(1,1)	1	ReLU
	Maxpool2	128 × 64 × 128	128 × 32 × 64	2 × 2	(0,0)	2	—
Down_ConvBlock_3	Conv_3_1	128 × 32 × 64	256 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Conv_3_2	256 × 32 × 64	256 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Conv_3_3	256 × 32 × 64	256 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Maxpool3	256 × 64 × 128	256 × 16 × 32	2 × 2	(0,0)	2	—
Down_ConvBlock_4	Conv_4_1	256 × 16 × 32	512 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Conv_4_2	512 × 16 × 32	512 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Conv_4_3	512 × 16 × 32	512 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Maxpool4	512 × 16 × 32	512 × 8 × 16	2 × 2	(0,0)	2	—
Down_ConvBlock_5	Conv_5_1	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
	Conv_5_2	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
	Conv_5_3	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
	Maxpool5	512 × 8 × 16	512 × 4 × 8	2 × 2	(0,0)	2	—
ST-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))						
ST-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	(0,0)	2	—
	Up_Conv_5_1	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
	Up_Conv_5_2	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
	Up_Conv_5_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	(0,0)	2	—
	Up_Conv_4_1	512 × 16 × 32	512 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Up_Conv_4_2	512 × 16 × 32	512 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Up_Conv_4_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	(0,0)	2	—
	Up_Conv_3_1	256 × 32 × 64	256 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Up_Conv_3_2	256 × 32 × 64	256 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Up_Conv_3_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	(0,0)	2	—
	Up_Conv_2_1	128 × 64 × 128	128 × 64 × 128	3 × 3	(1,1)	1	ReLU
	Up_Conv_2_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	(0,0)	2	—
	Up_Conv_1_1	64 × 128 × 256	64 × 128 × 256	3 × 3	(1,1)	1	ReLU
	Up_Conv_1_2	64 × 128 × 256	2 × 128 × 256	3 × 3	(1,1)	1	LogSoftmax

Abbreviations: ConvGRU, convolutional gated recurrent unit; ConvLSTM, convolutional long short-term memory; SCNN, spatial convolutional neural network; ST-RNN, spatial–temporal recurrent neural network; ReLU, Rectified Linear Unit.

*Two types of ST-RNN, that is, ConvLSTM and ConvGRU are tested.

**ST-RNN blocks are tested with one hidden layer or two hidden layers.

TABLE A2 Parameter settings for each layer of the UNet-based neural network

Layer		Input (channel × hight × width)	Output (channel × hight × width)	Kernel	Padding	Stride	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	(0,4)	1	ReLU
	SCNN_Up	64 × 1 × 256	64 × 1 × 256	1 × 9	(0,4)	1	ReLU
	SCNN_Right	64 × 128 × 1	64 × 128 × 1	9 × 1	(4,0)	1	ReLU
	SCNN_Left	64 × 128 × 1	64 × 128 × 1	9 × 1	(4,0)	1	ReLU
Down_ConvBlock_1	Maxpool1	64 × 128 × 256	64 × 64 × 128	2 × 2	(0,0)	2	—
	Conv_1_1	64 × 64 × 128	128 × 64 × 128	3 × 3	(1,1)	1	ReLU
	Conv_1_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	(0,0)	2	—
	Conv_2_1	128 × 32 × 64	256 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Conv_2_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	(0,0)	2	—
	Conv_3_1	256 × 16 × 32	512 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Conv_3_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	(0,0)	2	—
	Conv_4_1	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
	Conv_4_2	512 × 8 × 16	512 × 8 × 16	3 × 3	(1,1)	1	ReLU
ST-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))						
ST-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	(0,0)	2	—
	Up_Conv_4_1	1024 × 16 × 32	256 × 16 × 32	3 × 3	(1,1)	1	ReLU
	Up_Conv_4_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	(0,0)	2	—
	Up_Conv_3_1	512 × 32 × 64	128 × 32 × 64	3 × 3	(1,1)	1	ReLU
	Up_Conv_3_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	(0,0)	2	—
	Up_Conv_2_1	156 × 64 × 128	64 × 64 × 128	3 × 3	(1,1)	1	ReLU
	Up_Conv_2_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	(0,0)	2	—
	Up_Conv_1_1	128 × 128 × 256	64 × 128 × 256	3 × 3	(1,1)	1	ReLU
	Up_Conv_1_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	(0,0)	1	—

Abbreviations: ConvGRU, convolutional gated recurrent unit; ConvLSTM, convolutional long short-term memory; SCNN, spatial convolutional neural network; ST-RNN, spatial–temporal recurrent neural network; ReLU, Rectified Linear Unit.

*Similar to the SegNet-based network architecture, two types of ST-RNN, that is, ConvLSTM and ConvGRU, are tested.

**ST-RNN blocks are tested with one hidden layer or two hidden layers.