Efficient sequential neural network based on spatial-temporal attention and linear LSTM for robust lane detection using multi-frame images

This is the source code description for **Spatial-Temporal Attention Integrated Sequential Neural Network Model** which is used for robust lane detection using multi continuous image frames.

1. Background and Summary

Lane detection serves as a fundamental task for automated vehicles and Advanced Driver Assistance Systems. However, existing lane detection methods often fail to deliver the versatility of accurate, robust, and real-time compatible lane detection, especially under challenging driving scenes. Available vision-based methods in the literature frequently overlook critical regions of the image and their spatial-temporal salience regarding the detection results, leading to poor performance in peculiar difficult circumstances (e.g., serious occlusion, dazzle lighting). To address these limitations, this study introduces a novel spatial-temporal attention mechanism that can focus on key features of lane lines and exploit salient spatial-temporal correlations among continuous image frames to enhance the accuracy and robustness of lane detection. Under the standard encoder-decoder structure and with the implementation using common neural network backbones, efficient sequential neural network models are developed incorporating the proposed spatial-temporal attention mechanism. The developed models are trained and evaluated on three large-scale open-source datasets. Extensive experiments demonstrate the strength and robustness of the developed model outperforming available state-of-the-art methods across various testing scenarios. Furthermore, with the spatial-temporal attention mechanism, the developed sequential neural network models exhibit fewer parameters and reduced Multiply-Accumulate Operations (MACs) compared to baseline sequential models, highlighting their computational efficiency and real-world applicability.

2. Overall architecture

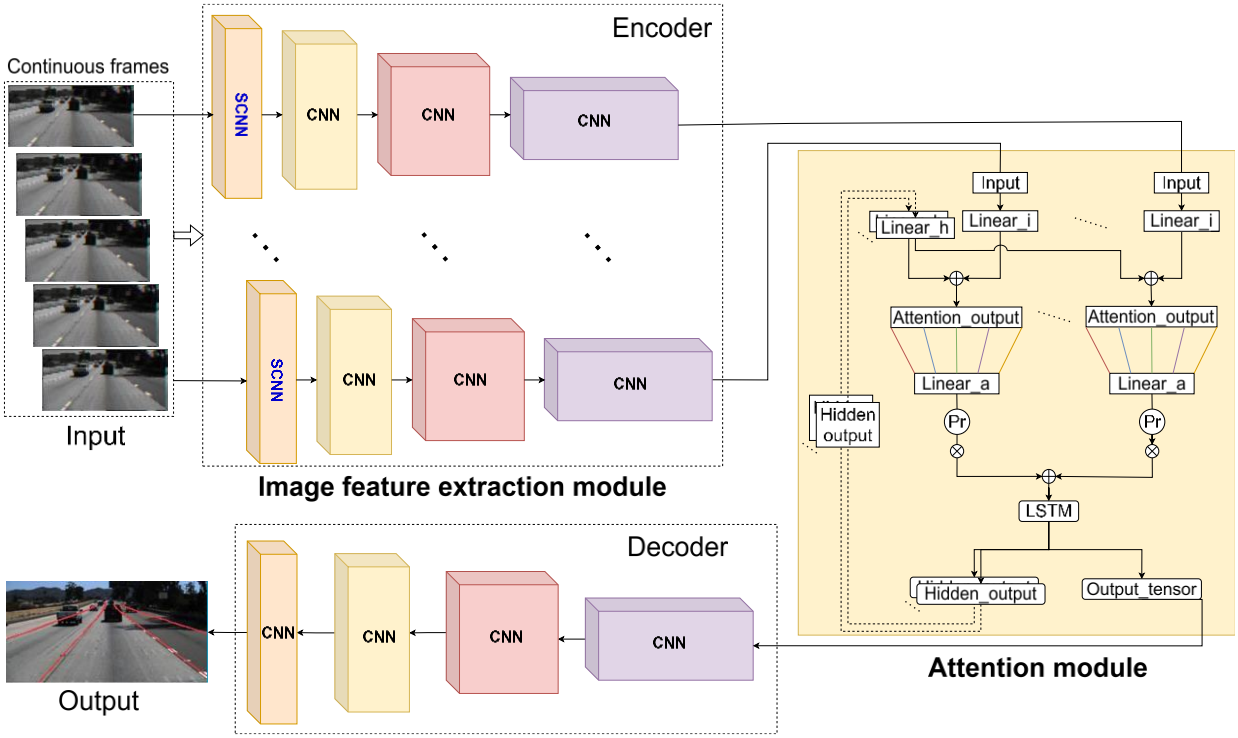

Fig. 1. The architecture of the proposed model.

An architecture overview of the proposed method is illustrated in Fig. 1.

3. Spatial-temporal attention mechanism

The proposed attention module is developed to mimic human visual cognitive attention which demonstrates the ability to focus on important parts and ignore minor parts. The attention module helps the neural network learn to focus on important frames and salient regions of each frame by assigning different weights to each image frame and particular regions of each frame. With the help of the embedded temporal feature extractor, e.g., LSTM or Gated Recurrent Unit (GRU), the attention module can also extract important temporal dependencies over the input consecutive image frames.

As illustrated in **Fig. 1**, the attention module is applied when the input image sequences are downsized and the features are extracted by a series of convolution layers in the encoder. The attention module integrates the extracted features from the encoder and the hidden outputs produced by the embedded temporal feature extractor, e.g., LSTM/GRU. The LSTM/GRUs' hidden outputs of the very last previous time step and the input feature maps at the current time step are combined using a set of attention weights. Activation of these weights can then be obtained to learn which image frame and which specific regions are important for the lane detection task. The weighted sum of input feature maps highlights the salient features, which are then processed by the temporal feature extractor to produce the output at the current time step and updated hidden state. All the attention weights can be trained simultaneously together with other neural network layer weights using the backpropagation mechanism. **Equations (1)-(13)** provide a formal mathematical description of the attention mechanism as described above.

The output of the final downsized convolutional block at time t for the n-th frame (i.e., timestep n within the image sequence) is denoted as $x_{down4}^{(t-N+n)}$, where $n = \{1, 2, , . . , N\}$, and N is the number of frames in the sequence, (in this implementation $N = 5$). The input sequence for the attention module is therefore defined as $\{x_{down4}^{(t-N+1)}, x_{down4}^{(t-N+2)}, . . , x_{down4}^{(t)}\}$. Please note there are two distinct temporal increments. The increment in n corresponds to processing the subsequent image in the input sequence; where an increment in time t reflects the real-world temporal progression, i.e., moving to the next input sequence. Then, within a certain selected sequence, the following computations are performed:

$$x^{(t-N+n)} = Conv\left(x_{down4}^{(t-N+n)},\ k_{in}\right) \tag{1}$$

$$z^{(t-N+n)} = \left(U \cdot x^{(t-N+n)}\right) + \left(H \cdot h^{(t)}\right) \tag{2}$$

$$w^{(t-N+n)} = softmax\left(W \cdot z^{(t-N+n)}\right) \tag{3}$$

$$\overline{x^{(t)}} = \sum_{n=1}^{N} w^{(t-N+n)} \cdot x^{(t-N+n)} \tag{4}$$

Here, "*Conv*" denotes the convolution operation, while "+" represents the element-wise addition operation. k_{in} is a convolution layer with a kernel of size of 1 × 1 and 1 channel (as indicated by *In_Attention_Conv_5_1* in **Table 1**). The matrices U, H, W are the learnable weights that can be configured as trainable vectors of size 1 × 1 or 1 × 128, or as a trainable fully connected layer of size 1 × 128. $x^{(t-N+n)}$ and $z^{(t-N+n)}$ represent the intermediate outputs. $w^{(t-N+n)}$ denotes attention weights obtained from softmax operation, and $\overline{x^{(t)}}$ is the attention-based weighted average output aggregating information across the sequence

$\{x_{down4}^{(t-N+1)}, x_{down4}^{(t-N+2)}, \ldots, x_{down4}^{(t)}\}$.

After processing the *N* images, and getting the weighted average $\overline{x^{(t)}}$ for the selected sequence, the following computations are carried out:

$$o^{(t)}, h^{(t+1)} = F(\overline{x^{(t)}},\ h^{(t)}) \tag{5}$$

$$x_{out} = Conv(o^{(t)},\ k_{out}) \tag{6}$$

where *F* is an embedded temporal feature extractor; $h^{(t)}$ is the hidden state vector initialized as $h^{(0)} = \mathbf{0}$ (zero vector) when $t = 0$ and $h^{(t)}$ will be updated with its new inheritor after the selected sequence is fully processed as in equation (5); $o^{(t)}$ is the output from *F*, which is then expanded to 512 channels by *outconv* layer k_{out}; k_{out} has a kernel size of 1 × 1 and 512 channels (indicated by *Out_Attention_Conv_5_2* in **Table 1**); x_{out} is the final output of the attention module which is then transferred to the decoder module.

The temporal feature extractor *F* can be LSTM or GRU. Take LSTM for an example, an LSTM unit is visualized in **Fig. 2**. Here, *i*, *f*, and *o* stand for input gate, forget gate, and output gate, respectively. The input (*IN*) is summed with a memory cell (C) and a new-memory cell ($\tilde{C}$). The output of $\tilde{C}$ is summed with a memory cell via forget gate (f). The output (OUT) is obtained from C via the activation function and output gate (o).

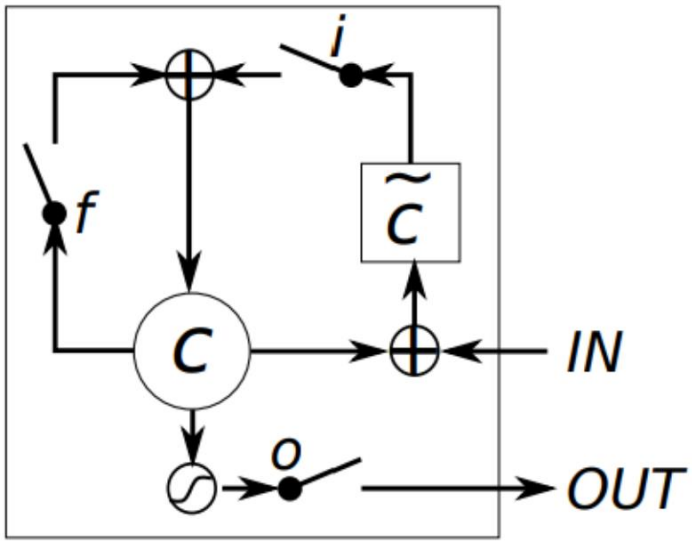

Fig. 2. An illustration of the Long Short Term Memory unit (Chung et al., 2014).

The key formulations of the LSTM are shown by the following equations:

$$f^{(t)} = \sigma\left(b^f + P^f\overline{x^{(t)}} + Q^f h^{(t)}\right) \tag{7}$$

$$i^{(t)} = \sigma\left(b^i + P^i\overline{x^{(t)}} + Q^i h^{(t)}\right) \tag{8}$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \tag{9}$$

$$\tilde{c}^{(t)} = g\left(b^c + P^c\overline{x^{(t)}} + Q^c h^{(t)}\right) \tag{10}$$

$$h^{(t+1)} = o^{(t)} \odot g(c^{(t)}) \tag{11}$$

$$o^{(t)} = \sigma\left(b^o + P^o x^{(t)} + Q^o h^{(t)}\right) \quad (12)$$

where $\overline{x^{(t)}}$ serves as the current input vector, h is the current hidden state vector as explained before, g is the typically hyperbolic tangent function, σ is the activation function, and $\odot$ is the Hadamard (element-wise) multiplication. b^f , P^f , Q^f are biases, input weights and recurrent weights for the forget gates; b^i, P^i, Q^i are biases, input weights and recurrent weights for the input gate; b^c , P^c , Q^c are biases, input weights and recurrent weights for the current value of the memory state $c^{(t)}$. $\tilde{c}^{(t)}$ is the internal candidate memory state. $o^{(t)}$ is the output at the output gate. b^o, P^o, Q^o are biases, input weights and recurrent weights for the output gate.

The attention model is implemented after the encoder module (to be specific, the fourth down sampling convolutional block, i.e., *Down_ConvBlock_4* in **Table 1**) and before the decoder module (to be specific, the first upsampling convolutional block, i.e., *Up_ConvBlock_4* in **Table 1**). Furthermore, one should notice that the proposed attention model is modular in nature and can be adopted with any network backbones not only UNet but also backbones such as SegNet (Badrinarayanan et al., 2017) and fully convolutional networks (Shelhamer et al., 2017).

In the implementation, depending on different settings of the learnable weights U, H, W in **Equations (2)-(3)**, three variants of the proposed attention module are developed and tested. They are temporal attention (Tem_Att, for short), spatial-temporal attention (ST_Att, for short), and spatial-temporal attention model with fully connected layers (STFC_Att, for short).

3.1 Temporal attention

The design of the spatial-temporal attention mechanism began with assessing the significance of each frame in a sequence for detecting lane markings in the current frame, which is implemented through the temporal attention (Tem_Att) module. In this module, the learnable weights of *U*, *H*, and *W* in equations (2)-(3) are trainable vectors and are illustrated by V_i, V_h and V_a in **Fig. 3** respectively. The three trainable vectors, each of size 1 × 1, dynamically adjust the contributions of input features, hidden state output, and the attention output based on the learned weights. Specifically, the input features $x^{(t-N+n)}$ are modulated by the weight vector V_i and combined with the hidden output multiplied by V_h through element-wise addition to construct a summed intermediate attention signal $z^{(t-N+n)}$, as described in equation (2). This attention signal is subsequently passed through a *softmax* activation function shown as 'Pr' in **Fig. 3** to compute the attention weights $w^{(t-N+n)}$, as defined in equation (3). These weights effectively prioritize the significance of each frame in the sequence.

Leveraging the LSTM unit (detailed in equations (7)–(12)), the hidden state $h^{(t)}$ contextualizes the input features by incorporating information from the entire sequence within the selected time window. The attention output $\overline{x^{(t)}}$ computed as a weighted combination of the input features $x^{(t-N+n)}$ and their respective attention weights $w^{(t-N+n)}$ (see Equation (4)), captures these temporal dependencies. This output is processed through the LSTM and a convolutional layer (as outlined in Equation (6)) to generate the module's final output x_{out}, which is subsequently passed to the decoder. When the three trainable vectors V_i, V_h and V_a are of size 1×1, this approach ensures that the model dynamically adapts its focus to relevant temporal features in the image sequence.

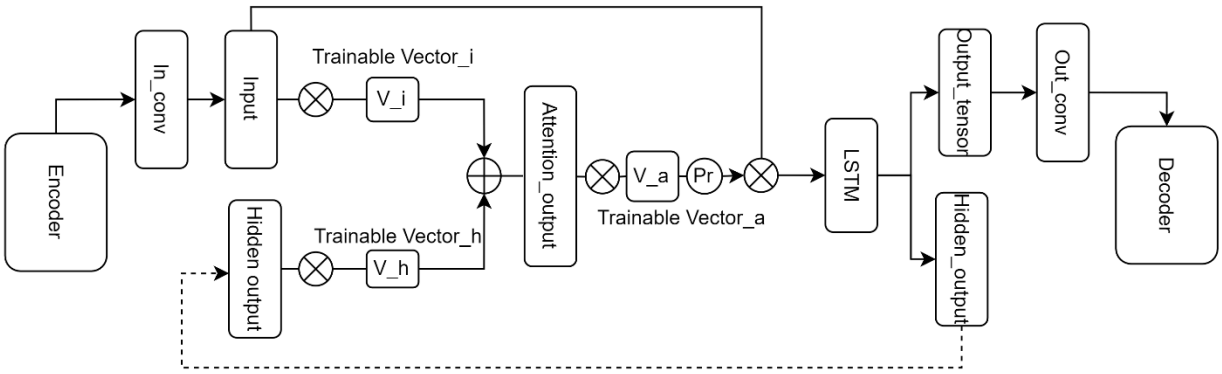

Fig. 3. An illustration of the temporal attention module (Tem_Att).

3.2 Spatial-temporal attention

Observations show that lane lines typically appear in specific regions within image frames, and certain features hold greater significance for accurate detection. To account for this, a spatial attention operation is applied to each frame, upgrading the Tem_Att module into the spatial-temporal attention (ST_Att) module. The ST_Att module introduces three learnable weight vectors, each of size 1×128, which are applied to the input feature matrix, the hidden state output, and the attention output at the current step. These weights with the size of 1×128 enable the module to prioritize important features within each frame. However, the ST_Att module does not account for the spatial relationships and dependencies between neighbouring feature maps, which are addressed in the subsequent STFC_Att module.

The structure of the ST_Att module is depicted in **Fig. 4**. While the workflow of ST_Att is similar to Tem_Att, the connections between the input features, hidden state outputs, and attention outputs, along with their respective weight matrices, follow a one-to-one mapping. These connections are illustrated with colour-coded lines in **Fig. 4**. Similar to Tem_Att, the attention weights are normalized to a range of 0 to 1 using a softmax activation function (denoted as 'Pr' in **Fig. 4**). The final attention output is processed through a convolutional layer before being passed to the decoder module. This mechanism ensures that the model emphasizes the most critical spatial features in each frame. When combined with the temporal modelling capabilities of the LSTM, it effectively leverages spatial-temporal information across image frames in the sequence, significantly enhancing the overall performance of lane detection.

3.3 Spatial-temporal attention with fully connected layers

The Spatial-Temporal Attention with Fully Connected Layers (STFC_Att) module builds upon the ST_Att module by incorporating a fully connected mechanism to enhance feature learning. Unlike the one-to-one connections in ST_Att, the STFC_Att module employs many-to-many connections, where each learnable weight matrix is multiplied with all values of the input feature map, as illustrated in **Fig. 5**. This many-to-many connection allows the model to extract spatial dependencies between feature maps within the same image frame while concurrently capturing temporal features and correlations across consecutive frames with the assistance of the LSTM's hidden outputs.

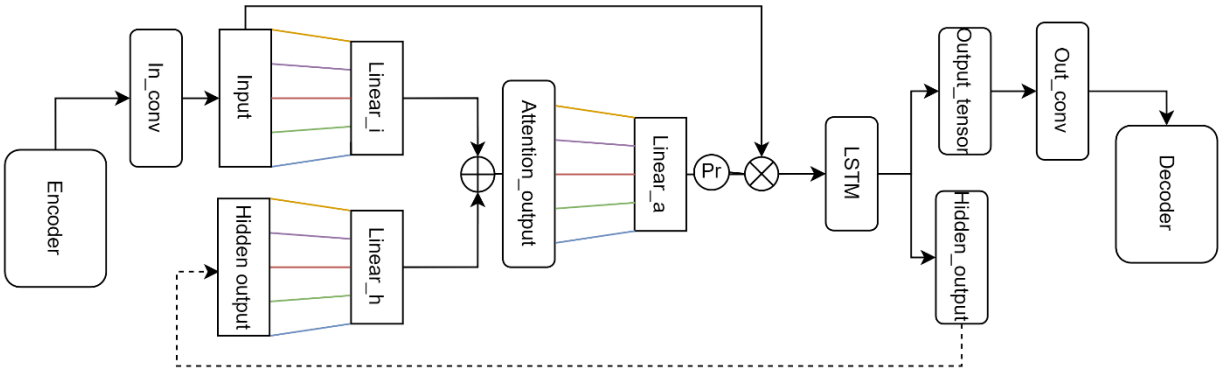

Fig. 4. An illustration of spatial-temporal attention module (ST_Att).

The structure of the STFC_Att module is depicted in **Fig. 5**, where different coloured lines represent the many-to-many connections between the input feature matrix $x^{(t-N+n)}$, the hidden state output $h^{(t)}$, and the attention output $\overline{x^{(t)}}$, along with their corresponding learnable weight matrices *U*, *H*, and *W*, denoted in **Fig. 5** as Linear_i, Linear_h, and Linear_a, respectively. Each of these matrices is implemented as a trainable fully connected layer of size 1×128. These weight matrices dynamically adjust the importance of both spatial and temporal features, ensuring a robust and comprehensive feature extraction.

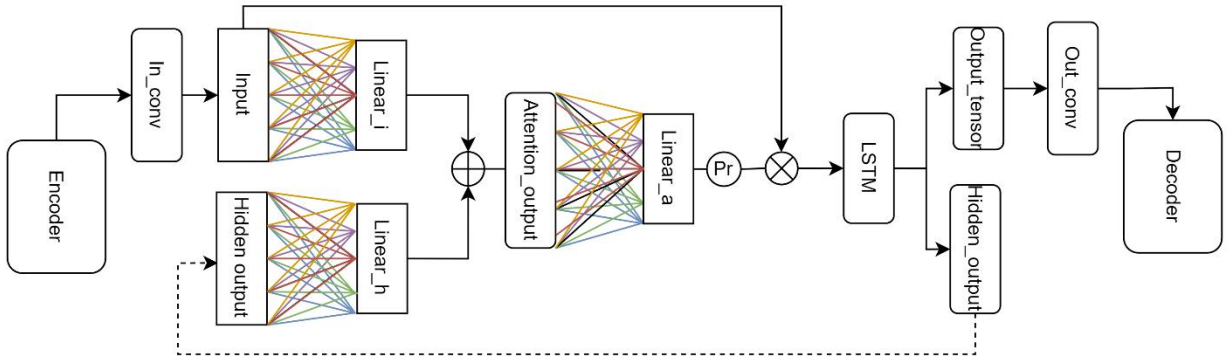

Fig. 5. An illustration of spatial-temporal attention module with fully connected layers (STFC_Att).

Similar to Tem_Att and ST_Att, in the STFC_Att module, the attention output, denoted as $\overline{x^{(t)}}$, is calculated using equation (4) as a weighted combination of the input features $x^{(t-N+n)}$ and their corresponding attention weights $w^{(t-N+n)}$. Subsequently, the robust attention output is processed through a linear layer of size 1×128 with many-to-many connections. This output is then scaled to a range of 0 to 1 using the softmax function (denoted as 'Pr' in **Fig. 5**). Finally, the processed attention outputs are passed through a convolutional layer, as outlined in equation (6), before being transferred to the decoder.

The key distinction between ST_Att and STFC_Att lies in their ability to capture spatial dependencies. While the ST_Att module focuses on weighting local spatial features within each frame, the fully connected mechanism in STFC_Att extends the network's capability by establishing interrelations between spatial features across the

entire frame and throughout the input image sequence. This enhancement allows the model to distinguish among spatial-temporal features more effectively and to focus its attention on the most relevant patterns. By integrating the fully connected spatial-temporal attention mechanism, the STFC_Att module significantly enhances the model's ability to detect lane markings in diverse driving scenarios by leveraging both spatial and temporal interdependencies.

3.4 Implementation details

1) Deep Neural Network Details: On the whole, as illustrated in **Fig. 1**, the proposed method adopts an "encoder-attention module-decoder" based sequence-to-one architecture. UNet (Ronneberger et al., 2015) is used as the neural network backbone, in which there are one In_ConvBlock and four consecutive down sampling convolutional blocks (i.e., Down_ConvBlock_1, Down_ConvBlock_2, Down_ConvBlock_3, Down_ConvBlock_4) in the encoder part, and four symmetrical upsampling convolutional blocks (i.e., Up_ConvBlock_4, Up_ConvBlock_3, Up_ConvBlock_2, Up_ConvBlock_1) in the decoder part. Between the encoder and the decoder, there is the attention module with a temporal feature extractor (e.g., LSTM) embedded. **Table 1** illustrates in detail the input and output sizes, as well as the parameters of each layer in the entire DNN.

2) Loss function: Vision-based lane detection can be considered as the pixel-wise binary classification problem, for which cross-entropy is a suitable loss function (Ho & Wookey, 2020). It is important to note that, in most cases, the pixels classified as "lanes" are far fewer than those classified as "no lanes" (i.e., the background), which makes it an unbalanced discriminative binary classification problem. Therefore, this study adopts the weighted cross-entropy as the loss function with two rescaling weights given to each class. The two weights for lane class and background class are set to the inverse proportion of the number of pixels in the two classes, e.g., there are fewer lane pixels than the background, so the weight of the lane class is larger. The adopted weighted binary cross-entropy loss function is illustrated by **Equation (13)**.

$$Loss = -\frac{1}{M}\sum_{m=1}^{M}\left[w_l * y_m * log\left(h_\theta(x_m)\right) + w_{nl} * (1\text{-}y_m) * log\left(1 - h_\theta(x_m)\right)\right] \quad (13)$$

where M is the number of training examples; w_l stands for the weight of lane class, while w_{nl} for the background class; y_m is the true target label for training example m; x_m is the input for training example m; and h_θ is the neural network model with weights θ.

3) Training details: Various variants of the developed neural network model, as well as selected baseline models, had been trained and tested on the Dutch national high-performance supercomputer cluster Lisa using four Titan RTX GPUs with the data trained parallelly using *torch.nn.DataParallel()* in PyTorch library. The input image size is set as 128 × 256 and the training batch size is set as 64. The learning rate is initially set to 0.01 with decay applied after each epoch. The Adam (Kingma & Ba, 2015), RAdam optimizer (Liyuan Liu et al., 2020) and Stochastic Gradient Descent (SGD) (Bottou, 2010) optimizers were all tested. Experiments demonstrated that SGD delivered the smallest loss in this study. Thus, SGD optimizer was chosen and the momentum term was applied.

Table 1

Neural Network architecture parameters.

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
Down_ConvBlock_1	Maxpool_1	64×128×256	64×64×128	2×2	(0,0)	2	---
	Down_Conv_1_1	64×64×128	128×64×128	3×3	(1,1)	1	ReLU
	Down_Conv_1_2	128×64×128	128×64×128	3×3	(1,1)	1	ReLU
Down_ConvBlock_2	Maxpool_2	128×64×128	128×32×64	2×2	(0,0)	2	---
	Down_Conv_2_1	128×32×64	256×32×64	3×3	(1,1)	1	ReLU
	Down_Conv_2_2	256×32×64	256×32×64	3×3	(1,1)	1	ReLU
Down_ConvBlock_3	Maxpool_3	256×32×64	256×16×32	2×2	(0,0)	2	---
	Down_Conv_3_1	256×16×32	512×16×32	3×3	(1,1)	1	ReLU
	Down_Conv_3_2	512×16×32	512×16×32	3×3	(1,1)	1	ReLU
Down_ConvBlock_4	Maxpool_4	512×16×32	512×8×16	2×2	(0,0)	2	---
	Down_Conv_4_1	512×8×16	512×8×16	3×3	(1,1)	1	ReLU
	Down_Conv_4_2	512×8×16	512×8×16	3×3	(1,1)	1	ReLU
Attention Module	In_Attention_Conv_5_1	512×8×16	1×8×16	1×1	---	1	---
	AttentionLayer_1	1×128*	1×128*	---	---	---	---
	AttentionLayer_2	1×128*	1×128*	---	---	---	---
	AttentionLayer_3	1×128*	1×128*	---	---	---	---
	LSTM	128	128	---	---	---	---
	Out_Attention_Conv_5_2	1×8×16	512×8×16	1×1	---	1	---
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	UpsamplingBilinear2D_3	128×32×64	128×64×128	2×2	(0,0)	2	---
	Up_Conv_2_1	256×64×128	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	UpsamplingBilinear2D_4	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	---

*This is an example of the spatial-*temporal attention* (*ST_Att*) module. Corresponding to three attention variants, parameters in AttentionLayer_1, AttentionLayer_2, and AttentionLayer_3 will be learnable vectors of size 1 × 1 for *Tem_Att*, learnable vectors of size 1 × 128 for *ST_Att*, or learnable vectors with many to many connections of size 1 × 128 for *STFC_Att*, respectively.

4. Experiments and results

To verify the effectiveness and robustness of the proposed model with the designed attention module, extensive experiments were carried out on three commonly used large-scale open-source datasets, i.e., TuSimple, tvtLANE (Zou et al., 2020), and LLAMAS (Behrendt & Soussan, 2019) datasets. Several DNN-based lane detection models, e.g., LaneNet (Neven et al., 2018), SCNN (Pan et al., 2018), Seg-Net (Badrinarayanan et al., 2017), UNet (Ronneberger et al., 2015), SegNet_ConvLSTM (Zou et al., 2020), and UNet_ConvLSTM (Zou et al., 2020), were selected as the baselines.

4.1 Test on tvtLANE and TuSimple Datasets

4.1.1 Datasets description

The original dataset of the *TuSimple Lane Detection Challenge* consists of 3,626 training and 2,782 testing one-second clips that are collected under different driving conditions. Each clip is extracted into 20 continuous frames, and only the last frame, i.e., the 20th frame, is labelled as the ground truth. Additionally, Zou et al. (2020) added the label of the 13th frame and augmented the TuSimple dataset with 1,148 additional clips (with also the 13th and 20th frames labelled) regarding rural road driving scenes collected in China. Moreover, rotation, flip, and crop operations are employed for data augmentation, and finally, a total number of (3,626 + 1,148) × 4 = 19,096 sequences were produced, among which 38,192 frames are labelled with ground truth.

For testing, there are 2,782 testing clips in the original TuSimple dataset. While in tvtLANE, there are two different testing sets, namely, Testset #1 which is based on the original TuSimple test set for normal driving scene testing, as well as Testset #2 which contains 12 challenging driving scenarios for testing challenging scenes and assessing the model robustness.

In the training phase, three different sampling strides, with an interval of 1, 2, and 3 frames respectively, were adopted to adapt to different driving speeds which also augment the training samples by three times, whereas in the test phase, the sampling stride was set as a fixed interval of 1 frame.

Detailed descriptions of the two datasets and sampling settings can be found in (Dong et al., 2023; Zou et al., 2020).

4.1.2 Qualitative evaluation

As the intuitive evaluation approach with visualization, in this subsection, qualitative lane detection results of different models are demonstrated in figure visualizations. The figure demonstrations are helpful to identify the strengths and weaknesses of the evaluated models and provide insights.

1)Results on tvtLANE testset #1: normal driving scene testing

Samples of the results from lane detection segmentation on tvtLANE testset #1 are shown in **Fig. 6**. The lane lines are segmented into white pixels, while the background is displayed in black pixels. Three proposed attention-based model variants and the baseline deep learning models were tested. Here in **Fig. 6**, all of the results are without post-processing, which also applies to all the visualizations and quantitative evaluations discussed later in this paper.

Qualitatively, the models should be able to a) correctly predict the number of lanes; b) accurately locate the lane lines in the segmentation image; c) segment the lanes in thin lines without blurs; d) keep proper continuity

without unexpected breaks in continuous lanes. Regarding these aspects, the proposed models with attention mechanisms all deliver good results, especially the STFC_Att based model indicated in the last row (i) which output the thinner lane lines with good continuity and fewer blurs. One may argue that it does not detect the correct number of lanes in the first two columns from the left. However, when zooming in for details, one can identify that the model correctly detects the left road boundary lanes which are too difficult and even not labelled in the ground truth. This defect with the dataset is also discussed in (J. Zhang et al., 2022).

Furthermore, in accord with previous studies (Dong et al., 2023; J. Zhang et al., 2022; Zou et al., 2020), models using multi-continuous image frames generally outperform models using a single frame, as the latter output thick lines with heavy blurs.

2) Results on tvtLANE testset #2: challenging scenes

According to **Fig. 7**, the proposed model is compared qualitatively with the baseline models when faced with some extremely challenging driving scenarios (tested on the tvtLANE testset #2). Involving a broad range of challenging situations, the testset #2 is a separated new dataset which is unseen during the training phase. It is observed that all the models do not perform well, especially regarding the 3rd column where there are vehicle occlusions and dirt road surfaces simultaneously. However, similar to norm scenes, the proposed attention-based models overall surpass baselines with thinner continuous lines and more correct locations and lane numbers. Typically, shown in the 4th column of **Fig. 7**, the STFC_Att-UNet_LSTM model demonstrates superior results detecting smooth clear lines with the correct number of lanes in the serious vehicle occlusion case, in which almost all the other models are defeated. This can be inferred by its capability of exploring spatial-temporal correlations among neighbouring pixels.

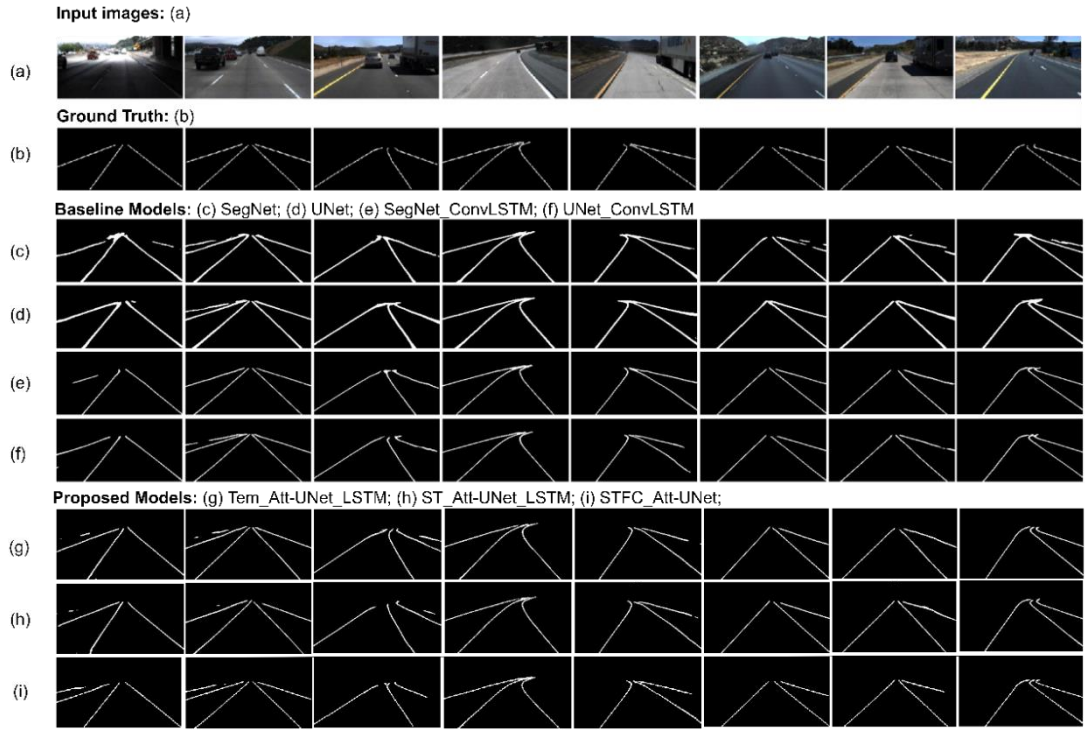

Fig. 6. Qualitative evaluation 1: Comparison of the results of lane detection on tvtLANE testset #1 (normal situations).

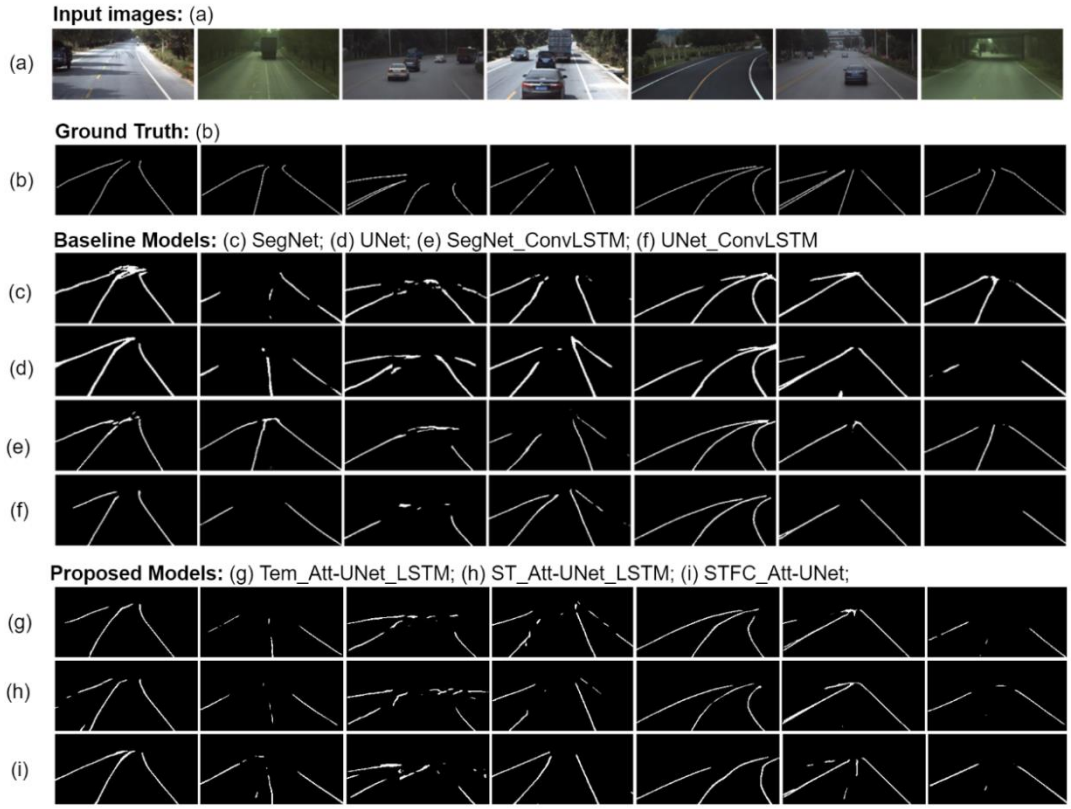

Fig. 7. Qualitative evaluation 2: Comparison of the lane detection results on tvtLANE testset #2 (challenging situations).

3) Results on TuSimple testing set

As mentioned before, the TuSimple testing set is similar to the tvtLANE testset#1, thus similar patterns are observed in **Fig. 8**. Compared with the baseline model UNet_ConvLSTM, the proposed models can detect more correct lane lines with fewer blurs.

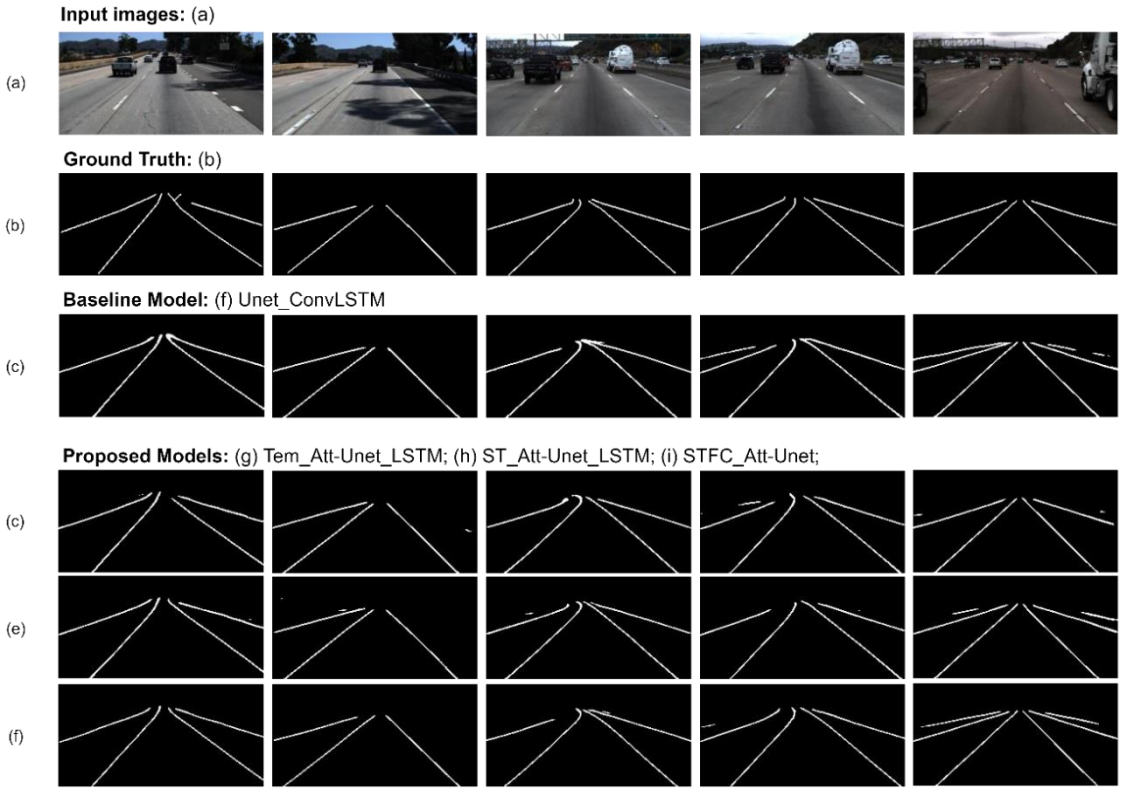

Fig. 8. Qualitative evaluation 3: Comparison of the lane detection results on the TuSimple test set.

4.1.3 Quantitative evaluation

1) Evaluation metrics: Treating the vision-based lane detection as a pixel-wise unbalanced two-class classification and discriminative segmentation task, and following the convention in previous studies (Dong et al., 2023; Lizhe Liu et al., 2021; Pan et al., 2018; H. Xu et al., 2020; J. Zhang et al., 2022; Zou et al., 2020), this study utilizes four commonly adopted evaluation criteria, i.e., accuracy, precision, recall, and F1-measure, to quantitatively verify the proposed models. The four criteria are illustrated in **Equations (14)~(17)**:

$$\text{Accuracy} = \frac{Truly\ Classified\ Pixels}{Total\ Number\ of\ Pixels} \tag{14}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive+False Positive}} \tag{15}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive+False Negative}} \tag{16}$$

$$\text{F1-measure} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision+Recall}} \tag{17}$$

where true positive correlates to the pixels that are accurately identified as lanes, false-positive indicates the number of background pixels that are incorrectly categorized as lanes, and false negative is for the number of lane pixels that were incorrectly categorized as background.

This study also provides the size of the model parameter, referred to as Params (M), as well as multiply-accumulate (MAC) operations, referred to as MACs (G), as indicators for estimating the computational complexity and capabilities of the models for real-time performance.

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

As shown in **Table 2**, when testing on tvtLANE testset#1, all the developed attention based models perform better than the baselines regarding F1-Measure, accuracy, and precision. This verifies the effectiveness of the proposed attention mechanism. The developed model STFC_Att-SCNN_UNet_LSTM (which will be discussed in detail in the ablation study) performs the best with the highest F1-Measure, accuracy, and precision. Furthermore, compared to the other two baseline models, i.e., UNet_ConvLSTM and SegNet_ConvLSTM, which also adopt multi frames as inputs, the developed models are all smaller in parameter size and with fewer MACs. This means that the developed models can deliver better results while using lower computation resources and with higher processing speed.

Table 2

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

		Test_Acc (%)	Precision	Recall	F1-Measure	MACs (G)	Params (M)
Models using **single image**	**Baseline Models**						
	U-Net	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 image frames**	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						
	Tem_Att-UNet_LSTM	98.08	0.877	0.936	0.906	44.7	13.5
	ST_Att-UNet_LSTM	98.09	0.879	0.941	0.909	44.8	13.5
	STFC_Att-UNet_LSTM	**98.14**	**0.887**	0.941	**0.911**	44.9	13.5
	*STFC_Att-SCNN_UNet_LSTM***	***98.20***	***0.906***	*0.936*	***0.921***	*68.9*	*13.7*

* Results reported in (J. Zhang et al., 2022).

** Model variant used for ablation study.

Tem_Att-UNet_LSTM means the temporal attention based model using the UNet_LSTM network backbone. This naming rule also applies to other models.

3) Quantitative performance comparison on tvtLane testset #2 (challenging scenes)

For testing model robustness, the developed models were also evaluated and verified on the brand-new dataset, namely the tvtLANE testset #2, which contains 12 challenging scenes.

As shown in **Table 3**, in terms of precision, ST_Att-UNet_LSTM performs the best in *bright*, *curve*, and *urban* scenes, while STFC_Att-UNet_LSTM performs the best in *occluded*, *shadow* and *tunnel* scenes. Therefore, they dominate half of the 12 challenging scenes.

High precision means the model is more strict for the pixels to be classified as lane lines, i.e., fewer False Positives. This is crucial for the vehicles' localizing lanes. However, being too strict might result in more False Negatives, then a lower recall ratio, and then a worse F1-measure. This is why the developed models are not good in terms of F1-measure. Furthermore, it is witnessed that during the training process, all the models

obtained higher recalls and lower precisions at the beginning. Then as the training went on, the recalls decreased while the precisions rose. This general pattern applies to all models. With this, one can infer that a higher precision is more important. All these demonstrate the developed models' robustness over challenging scenes.

Table 3
Model quantitative performance comparison on tvtLane testset #2 (12 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**
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
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
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**
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
Tem_Att-UNet_LSTM	**0.8430**	**0.8909**	0.7732	0.5740	0.8322	0.4692	0.4567	0.8358	0.8090	0.9244	0.7893	0.8046
ST_Att-UNet_LSTM	0.7938	0.8743	**0.8013**	0.7014	**0.8894**	0.5215	**0.4935**	0.8290	0.8517	0.9286	0.7516	0.8218
STFC_Att-UNet_LSTM	0.8239	0.8782	0.7646	**0.7031**	0.8871	0.5295	0.4848	0.7354	0.9023	**0.9395**	**0.8794**	0.7542
F1-MEASURE												
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
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
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
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**
Tem_Att-UNet_LSTM	**0.8933**	0.8657	0.8123	0.6513	0.8306	0.3530	0.5263	0.8290	0.7039	0.5338	0.5225	0.5226
ST_Att-UNet_LSTM	0.8548	0.8977	0.8253	0.7293	0.8254	0.3627	0.5543	**0.8369**	0.7480	0.6197	0.5522	0.5363
STFC_Att-UNet_LSTM	0.8690	0.9059	0.8314	**0.7456**	0.8086	0.3660	0.5277	0.7715	0.7329	0.6543	0.6471	0.5852

4) Performance and comparisons on TuSimple testing set

The aforementioned TuSimple testing set has similar but more testing samples compared to the tvtLANE testset#1. Regarding the quantitative results on the TuSimple testing set, as demonstrated in **Table 4**, the proposed STFC_Att-UNet_LSTM obtains the best F1-measure, the best precision, and the second to best accuracy (i.e., 98.20% only a bit lower than the best of 98.22%). Although UNet_ConvLSTM shows the best accuracy, it is worth noting that its MACs and parameter size are much larger than the proposed models. In this case, one can conclude that the developed models with lower computational complexities are robust with good results on the TuSimple testing set.

Table 4

Model quantitative performance comparison on *TuSimple* testing set

		Test_Acc (%)	Precision	Recall	F1-Measure	MACs (G)	Params (M)
Models using **continuous image frames**	**Baseline Models**						
	SegNet_ConvLSTM*	97.96	0.852	0.964	0.901	217.0	67.2
	UNet_ConvLSTM*	**98.22**	0.857	0.958	0.904	69.0	51.1
	UNet_DoubleConvGRU*	98.04	0.875	0.953	0.912	---	13.4
	Proposed Models						
	Tem_Att-UNet_LSTM	98.05	0.876	0.923	0.899	44.7	13.5
	ST_Att-UNet_LSTM	98.14	0.881	0.925	0.902	44.8	13.5
	STFC_Att-UNet_LSTM	***98.20***	**0.886**	0.950	**0.917**	44.9	13.5

* Results reported in (J. Zhang et al., 2022).

4.2 Test on LLAMAS dataset

4.2.1 Datasets description

To further verify the robustness of the proposed method, the LLAMAS dataset (Behrendt & Soussan, 2019) is adopted to train, validate, and test different models. Consisting of a total of 100,042 images, LLAMAS is one of the largest open-source lane marker datasets. Among the 100,042 images, 79,113 of them are used for training with labelled ground truth, while 20,929 of them were originally used for testing with no corresponding labels. To still follow the proposed end-to-end supervised learning pipeline and make it comparable with the previous work (J. Zhang et al., 2022), this study follows the processes described in (J. Zhang et al., 2022) utilizing only the labelled 79,113 images and dividing them into two groups. To be specific, 58,269 images were used for training, and 20,844 images were used for testing. More details about the LLAMAS dataset can be found in (Behrendt & Soussan, 2019; J. Zhang et al., 2022).

4.2.2 Qualitative evaluation

Limited by computational resources and time, this study only trained ST_Att-UNet_LSTM and STFC_Att-UNet_LSTM models on the LLAMAS dataset. **Fig. 9** provided the qualitative visualization results of ST_Att-UNet_LSTM for testing on the LLAMAS dataset. In the top row, the predicted lane lines are shown in red colour, and in the bottom row, the predicted lane lines are segmented with white pixels under black background. As shown, the lane lines in LLAMAS are labelled in a different way using dash lines, which makes it much more

challenging. Qualitatively, from the visualization, one can observe that there are very few false positives and the lane lines are generally predicted accurately.

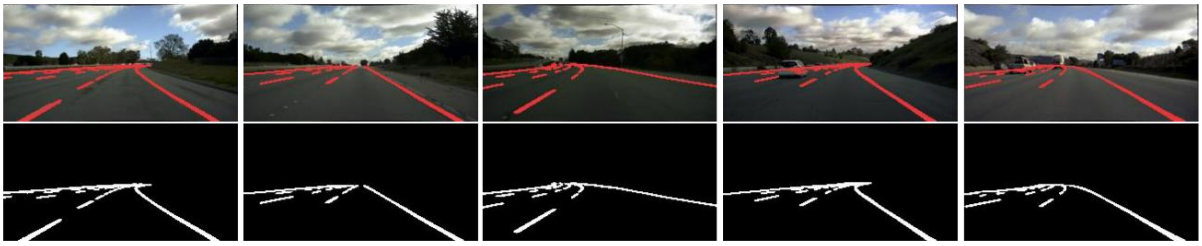

Fig. 9. Qualitative evaluation 4: Lane detection results on the LLAMAS dataset.

4.2.3 Quantitative evaluation

To quantitatively evaluate the model performances on the LLAMAS dataset, except for the aforementioned precision and recall, similar to (Behrendt & Soussan, 2019; J. Zhang et al., 2022), average precision (AP) was also adopted. AP is the mean of the weighted precision scores at different thresholds. The weights are the differences in recalls from the prior tested thresholds. To be clear, AP is illustrated in **Equation (18)**

$$\mathrm{AP} = \sum_{p=1}^{U} \sum_{q=1}^{V+1} (Precision_p * \Delta Recall_q) \tag{18}$$

where U means the total number of the tested image frames; V means the number of pixel samples for a single image; $\Delta Recall$ represents the difference between the *Recall* values of two consecutive samples. The variables p and q are subscripts to number the samples. In the implementation, similar to (J. Zhang et al., 2022), this study sets $Recall_0$ to 0, $Precision_0$ to 1, and the variable V to 100.

The quantitative results are demonstrated in **Table 5**.

Table 5.

Model quantitative performance comparison on the LLAMAS dataset

	Average Precision (AP)	**Precision**	**Recall**
UNet_Double_ConvGRU*	0.**8519**	0.6162	0.6163
SegNet ConvLSTM*	0.8500	0.5487	**0.6839**
UNet_ConvLSTM*	0.8510	0.5857	0.6558
ST_Att-UNet_LSTM	0.7106	0.6253	0.6584
STFC_Att-UNet_LSTM	0.7141	**0.6317**	0.6413

* Results reported in (J. Zhang et al., 2022).

As shown in **Table 5**, the STFC_Att-UNet_LSTM model provides the best corner precision when testing on the LLAMAS dataset. This is an indication that the model delivers a lower number of false positives, which, as discussed before, is more crucial for lane localization. It also obtains a comparable corner recall. Furthermore, it is worth noting that both the proposed models maintain a better balance among the three evaluation metrics although they do not perform well in average precision. To sum up, the developed models' robustness on the LLAMAS dataset is demonstrated with competitive quantitative and qualitative detection results.

4.3 Qualitative test on unlabeled Netherlands lane dataset

To further verify the developed models' robustness in handling new and challenging driving scenes, the unlabeled Netherlands lane dataset was adopted for qualitative testing. This dataset covers a wide range of driving situations in the Netherlands, some of which are very challenging.

Fig. 10. Qualitative evaluation 5: Lane detection results on unlabeled Netherlands lane dataset.

Fig. 10 shows the lane detection results of ST_Att-UNet_LSTM, which is only trained on the LLAMAS dataset. Even without any supervised training on the unlabeled Netherlands lane dataset, the proposed model demonstrates excellent transfer capabilities by clearly detecting lane line numbers and locations. Furthermore, the model can correctly identify whether the lanes are continuous or dashed lanes. The good performance can be attributed to that the developed ST_Att-UNet_LSTM with spatial-temporal attention module can aggregate rich valuable context information to focus on generalized information and salient regions in both one image and the continuous image frames. This qualitative testing further verifies the robustness of the developed model.

4.4 Post-explanation of attention mechanism by visualization

To explain how the proposed attention mechanism works, in this subsection, a case study with visualization is provided. Consider a vehicle running under a bridge as shown in **Fig. 11**. The top row shows the original continuous image frames, with the 6^{th} image just copying the 5^{th} one as the current frame for the detection task. In the bottom row, the first 5 images are the attention activation visualizations for the continuous image frames, and the 6^{th} image is the lane detection result. As shown in this case, the bridge casts shadows on a part of the road on the left side and the lane lines are not visible there. Also, on the right side, parts of the lanes are occluded by the front vehicle. The developed spatial-temporal based neural network model, therefore, focuses on features from previous frames to predict where the lane lines are located even when the lane lines are not visible in the current frame (the 5^{th} image in the top row). As shown, in frames 1, 2, 3 and 4 the lane lines are visible partially. The spatial-temporal based neural network model can detect points and regions of interest (illustrated by the light green colour) among these continuous frames and "memorize" the locations of the lane lines. Due to the continuous nature of the lane lines, the network will predict the lane line in the 5^{th} frame making use of the "memorized" information of the previous frames. This ability to memorize salient spatial-temporal correlations among continuous frames can also contribute to other adverse challenging situations.

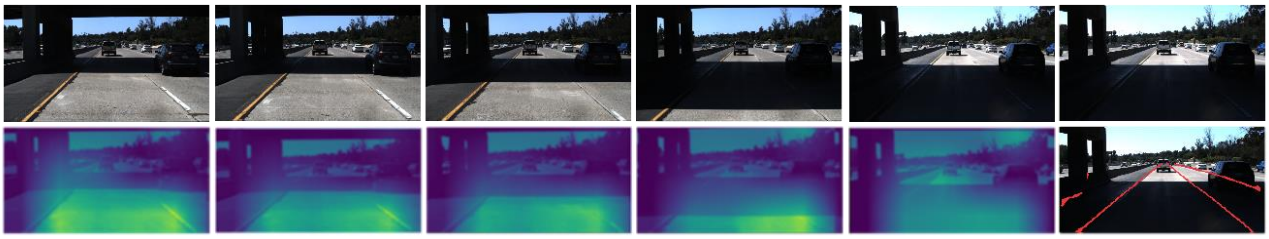

Fig. 11. Visualization of the case study: lane detection under a bridge with shadows and occlusion.

5. How to use the codes

(1) Download tvtLANE Dataset:

You can download this **dataset** from the link in the '**Dataset-Description-v1.2.pdf**' file.

BaiduYun：https://pan.baidu.com/s/1lE2CjuFa9OQwLIbi-OomTQ passcodes：tf9x

Or

Google Drive:
https://drive.google.com/drive/folders/1MI5gMDspzuV44lfwzpK6PX0vKuOHUbb_?usp=sharing

The **pretrained model** is also provided in the "**/model**" folder, named as 98.48263448061671_RAd_lr0.001_batch70_FocalLoss_poly_alpha0.25_gamma2.0_Attention_UNet_LSTM.pth .

(2) Set up

Requirements

PyTorch 0.4.0

Python 3.9

CUDA 8.0

Preparation

Data Preparation

The dataset contains 19383 continuous driving scenes image sequences, and 39460 frames of them are labeled. The size of images is 128*256.

The training set contains 19096 image sequences. Each 13th and 20th frame in a sequence are labeled, and the image and their labels are in “clips_13(_truth)” and “clips_20(_truth)”. All images are contained in “clips_all”.

Sequences in “0313”, “0531” and “0601” subfolders are constructed on TuSimple lane detection dataset, containing scenes in American highway. The four “weadd” folders are added images in rural road in China.

The testset has two parts: Testset #1 (270 sequences, each 13th and 20th image is labeled) for testing the overall performance of algorithms. The Testset #2 (12 kinds of hard scenes, all frames are labeled) for testing the robustness of algorithms.

To input the data, we provide three index files(train_index, val_index, and test_index). Each row in the index represents for a sequence and its label, including the former 5 input images and the last ground truth (corresponding to the last frame of 5 inputs).

The dataset needs to be put into a folder with regards to the location in index files, (i.e., txt files in "./data/". The index files should also be modified add cording to your local computer settings. If you want to use your own data, please refer to the format of our dataset and indexes.

(3) Training

Before training, change the paths including "train_path"(for train_index.txt), "val_path"(for val_index.txt), "pretrained_path" in **config_Att.py** to adapt to your environment.

Choose the models (**UNet_ConvLSTM** | SCNN_UNet_ConvLSTM | SCNN_UNet_Attention) as the default one which is also indicated by *default='UNet-ConvLSTM'* thus you do not need to make change for this. And adjust the arguments such as *class weights* (now the weights are set to fit the tvtLANE dataset), *batch size*, *learning rate*, and *epochs* in **config_Att.py**. You can also adjust other settings, e.g., optimizer, check in the codes for details.

Then simply run: **train.py**. If running successfully, there will be model files saved in the "./model" folder. The validating results will also be printed.

(4) Test

To evaluate the performance of a trained model, please select the trained model or put your own models into the "./model/" folder and change "pretrained_path" in **test.py** according to the local setting, then change "test_path" to the location of *test_index.txt*, and "save_path" for the saved results.

Choose the right model that would be evaluated, and then simply run: **test.py**.

The quantitative evaluations of Accuracy, Precision, Recall, and F1 measure would be printed, and the lane detection segmented results will be saved in the "./save/" folder as pictures.

6. Authors

Yongqi Dong (y.dong-4@tudelft.nl), Sandeep Patil, Haneen Farah, Hans Hellendoorn

Reference

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

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

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling 1–9.

Dong, Y., Patil, S., van Arem, B., Farah, H., 2022. A hybrid spatial–temporal deep learning architecture for lane detection. Comput. Civ. Infrastruct. Eng. 1–20. https://doi.org/10.1111/mice.12829

Ho, Y., Wookey, S., 2020. The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2962617

Kingma, D.P., Ba, J.L., 2015. Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

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.

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.

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

Shelhamer, E., Long, J., Darrell, T., 2017. Fully Convolutional Networks for Semantic Segmentation. IEEE Trans. Pattern Anal. Mach. Intell. https://doi.org/10.1109/TPAMI.2016.2572683

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.

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

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