# Robust Lane Detection through Self Pre-training with Masked Sequential Autoencoders and Fine-tuning with Customized PolyLoss

This is the source code of the paper:

R. Li and Y. Dong, "Robust Lane Detection Through Self Pre-Training With Masked Sequential Autoencoders and Fine-Tuning With Customized PolyLoss," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 14121-14132, Dec. 2023, doi: 10.1109/TITS.2023.3305015.

keywords: {Lane detection; Feature extraction; Neural networks; Image reconstruction; Image segmentation; Self-supervised pre-training; Masked sequential autoencoders; PolyLoss; Deep neural network; Deep learning; Computer vision}

In this document, we provide the dataset and the model description.

# Network Architecture

In this study, three neural network models, i.e., UNet_ConvLSTM, SCNN_UNet_ConvLSTM, and SCNN_UNet_Attention are tested.

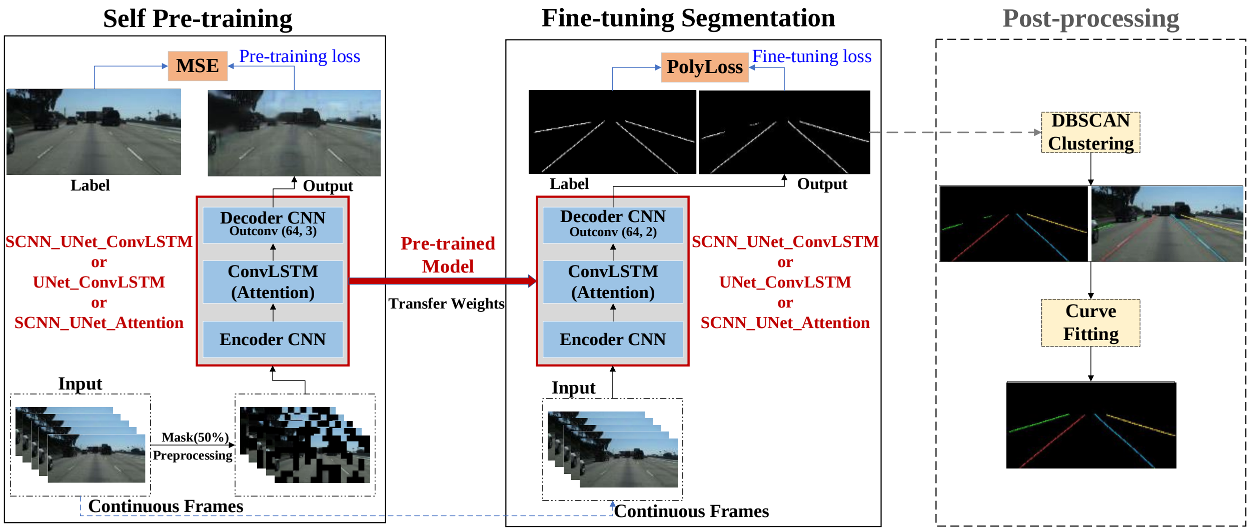

Fig. 1. The framework of the proposed pipeline.

# Some Results

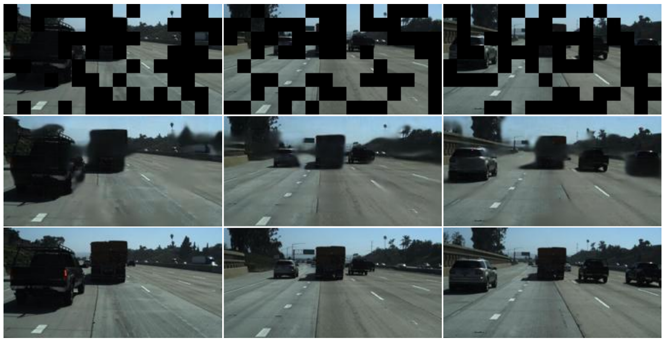

Fig. 2. Visualization of the reconstructing results in the pre-training phase.

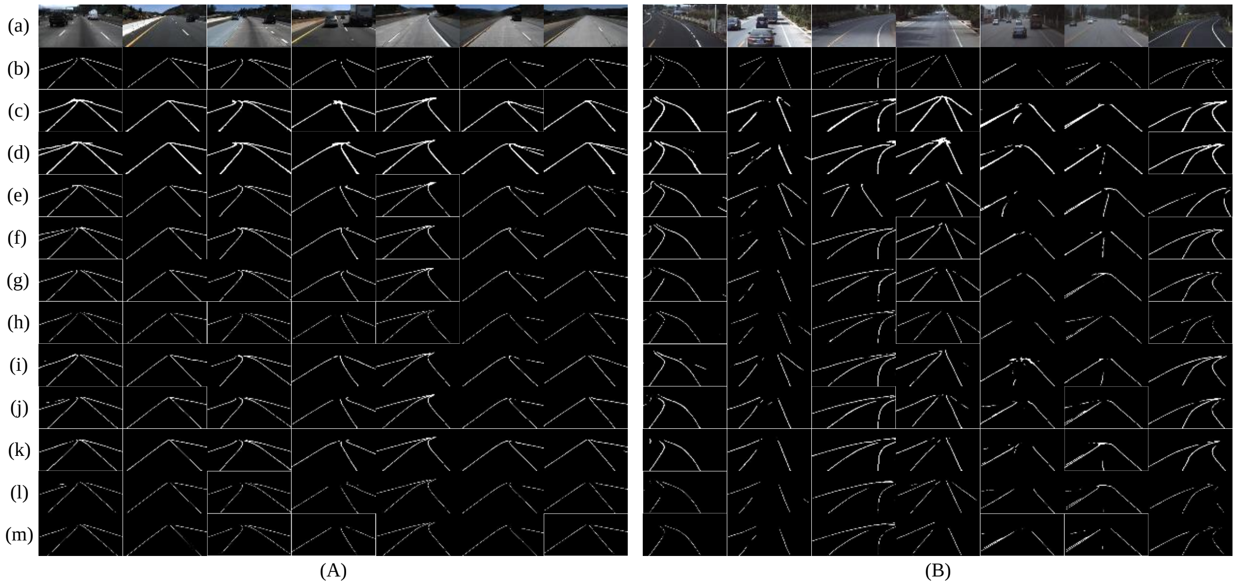

Fig. 3. Qualitative visual comparison of the lane detection results testing on tvtLANE test set #1 (normal) (A) and tvtLANE test set #2 (challenging) (B). All results in the figure are without post-processing. (a) Original input images; (b) Ground truth; (c)~(l) are the lane detection results corresponding to the models: (c) SegNet, (d) UNet, (e) SegNet_ConvLSTM [1], (f) UNet_ConvLSTM, (g) UNet_ConvLSTM_CE**, (h) UNet_ConvLSTM_PL**, (i) SCNN_SegNet_ConvLSTM [10], (j) SCNN_UNet_ConvLSTM, (k) SCNN_UNet_ConvLSTM_ CE**, (l) SCNN_UNet_ConvLSTM_PL**, (m) SCNN_UNet_Attention_PL**. (Note: CE and PL are short for weighted cross entropy loss and PolyLoss respectively, while ** means the model is pre-trained with the proposed self pre-training method.)

# tvtLANE Dataset

## Description: (adapted from https://github.com/qinnzou/Robust-Lane-Detection)

This dataset contains 19383 image sequences for lane detection, and 39460 frames of them are labeled. These images were divided into two parts, a training dataset contains 9548 labeled images and is augmented by four times, and a test dataset has 1268 labeled images. The size of images in this dataset is 128*256 pixels.

+ Training set:

- Data augmentation:

The training set is augmented. By flipping and rotating the images at three degrees, the data volume is quadrupled. These augmented data are separated from the original training set, which is named by “origin”. “f” and “3d” after “-” are represented for flipping and rotation. Namely, the “origin- 3df” folder is the rotated and flipped training set.

- Data construction:

The original training set contains continuous driving scene images, and they are divided into image sequences by twenty frames per second. All images are contained in “clips_all”, and there are 19096 sequences for training. Each 13th and 20th frame in a sequence are labeled, and the 38192 image and their labels are in “clips_13(_truth)” and “clips_20(_truth)”.

The original training dataset has two parts. Sequences in “0313”, “0531” and “0601” subfolders are constructed on the TuSimple lane detection dataset, containing scenes in American highways. The four “weadd” folders are added images in rural road in China.

+ Test set:

  - Testset #1:

The normal test set, named Testset #1, is used for testing the overall performance of algorithms. Sequences in “0530”, “0531” and “0601” subfolders are constructed on the TuSimple lane dataset. 270 sequences are contained, and each 13th and 20th image is labeled.

  - Testset #2:

Testset #2 is used for testing the robustness of algorithms. 12 kinds of hard scenes for human eyes are contained. All frames are labeled.

## Using:

Index files are provided. For detecting lanes in continuous scenes, the input size is 5 in the implementation. Thus, the former images are additional information to predict lanes in the last frame, and the last (5th) frame is the one with labeled ground truth.

We use different sampling strides to get 5 continuous images, as shown in the paper. Each row in the index files represents a sequence and its label for training. Refer to (https://github.com/qinnzou/Robust-Lane-Detection/blob/master/LaneDetectionCode/save/result/lane3.png)

## Download:

You can download this **dataset** from

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

Or

Google Drive:

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

You can also download the **pre-trained model** from the following link,

BaiduYun： https://pan.baidu.com/s/1ioOFRj6wZzORl6i5c73vCw With passcodes： y3sl

Or

Google Drive:

https://drive.google.com/drive/folders/1wF5m4GUEAgLWx3yb0GB09WOjNq2i3OZX?usp=drive_link

# Set up

## Requirements

PyTorch 1.10.2

Python 3.9

CUDA 8.0

cuDNN 11.1

## Preparation

### Data Preparation

The tvtLANE dataset contains 19383 continuous driving scenes image sequences, and 39460 frames of them are labeled. The size of the 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 highways. The four “weadd” folders are added images in rural road in China.

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

To input the data, the authors provide three index files (train_index, val_index, and test_index). Each row in the index represents 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 tvtLANE dataset can be downloaded and put into "./LaneDetectionCode/data/". If you want to use your own data, please refer to the format of the dataset and corresponding index files.

## Pretraining

Change the paths including "train_path"(for train_index.txt), "val_path"(for val_index.txt) in **config.py** to adapt to your environment.

Choose the models(UNet_ConvLSTM or Attention) and adjust the arguments such as class weights, batch size, and learning rate in config.py.

You should run pretrain.py at first to acquire the pre-trained model for the training phase.

The pre-trained model will save in the ./**model_pretrain** folder.

## Training

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

Choose the models (UNet_ConvLSTM | SCNN_UNet_ConvLSTM | SCNN_UNet_Attention) and adjust the arguments such as class weights, batch size, and learning rate in config.py.

Then simply run: train.py

## Test

To evaluate the performance of a trained model, please put the trained model listed above or your own models into "./LaneDetectionCode/model/" and change "pretrained_path" in test.py at first, then change "test_path" for 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 result pictures will be saved in "./LaneDetectionCode/save/test/".

# Citation:

Please cite our paper if you use this code or data in your own work:

```
@ARTICLE{10226453,
  author={Li, Ruohan and Dong, Yongqi},
  journal={IEEE Transactions on Intelligent Transportation Systems},
  title={Robust Lane Detection Through Self Pre-Training With Masked Sequential Autoencoders and Fine-Tuning With Customized PolyLoss},
  year={2023},
  volume={24},
  number={12},
  pages={14121-14132},
  doi={10.1109/TITS.2023.3305015}}
```

# Copy Right:

This dataset was collected for academic research only.

# Contact:

For any problem with this code implementation, please contact Ruohan Li (ruohanli373@gmail.com) or Yongqi Dong (qiyuandream@gmail.com).