CS230

Freeway Lane Detection using a Semantic Segmentation Approach

Junjie Lou
Stanford University
julou@stanford.edu

Xiangbing Ji
Stanford University
xji1994@stanford.edu

Zhengxun Wu
Stanford University
wukey92@stanford.edu

Abstract

Lane detection is to detect lanes on the road and provide the accurate location and shape of each lane. A robust and consistent lane detection engine helps to guide vehicles and could be used in driving assistance system [1]. Traditional lane detection methods heavily rely on hand-engineered features, which usually lack scalability and are prone to road environment variations [3]. Modern techniques leverage the advance of deep learning but usually come with sophisticated network architectures to adapt the uniqueness of lane detection. In this project, we treated lane detection as a binary semantic segmentation problem and proposed a simple encode-decoder architecture for lane detection. The model is a simplified version of SegNet [2] and we applied our model on the TuSimple Dataset. The results showed that we achieved an accuracy of 84.79%, which is comparable to LaneNet's 84.96% accuracy. (a state-of-the-art model that won 4th prize on TuSimple Dataset Challenge [3])

1 Introduction

Modern cars are incorporating an increasing number of driver assist features, among which automatic lane keeping. Despite being widely used, lane detection is still quite challenging due to several unique properties of lanes. Firstly, the long, curved lane shapes are hard to be captured by bounding boxes, thus making lane detection a hard problem for objection detection models like YOLO; Secondly, The inadequacy of distinctive features makes lanes tend to be confused by other objects with similar local appearance [4]. Thirdly, the inconsistent number of lanes on a road as well as diverse lane line patterns, e.g. solid, broken, single, double, merging, and splitting lines further impede the performance. In this project, we narrow down the problem to just freeway lane detection. The motivation is that we assume freeway lane detection is a much easier problem compared with generic lane detection while freeway driving actually consists of a great portion of a person's total driving time, especially for long distance driving. Therefore, the marginal cost for achieving generic lane detection is too high. To justify our assumption, we compared two different datasets: TuSimple Dataset and CULane Dataset. The TuSimple Dataset is a freeway only dataset while the CULane Dataset includes lots of local roads. Per Fig1, in TuSimple Dataset, the lanes are obvious for humans while local roads in CULane Dataset contain much more noises (pedestrians, bikes, street signs) and more variation of different lanes. Sometimes, it is even hard for humans to tell where the lane is thus making generic lane detection a very hard problem for models. Given the relatively lower difficulty level of freeway only lane detection, we think it is possible to tackle the problem end to end 1) without using hand-engineered features, heuristics and post-processing. 2) use a simpler network architecture compared with some state-of-the-art architectures like LaneNet [4] and Spatial CNN [5]. Our solution is tackling freeway lane detection using a semantic segmentation approach. There are

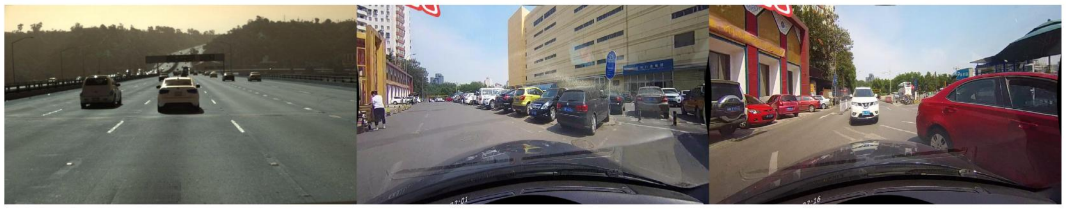

Figure 1: TuSimple Dataset (freeway) vs CULane Dataset (local)

only two classes in this semantic segmentation: lanes VS non-lane. So the input to our model is an image and the output of our model is an image with the same size. In the output image, pixels with value 1 denote this pixel belongs to a lane in the input image; while pixels with value 0 denote the pixel does not belong to any lanes in the input image.

2 Related work

2.1 Semantic Segmentation

SegNet is proved to be a very effective algorithm for semantic segmentation and based on the encoder-decoder architecture [2]. FCN [6] and PSPNet [7] are two other widely used semantic segmentation algorithms. They are also based on the encoder-decoder architecture but have more inner stacking and connections.

2.2 Lane Detection

Pan et al proposed spatial CNN, where convolutions are done slice by slice within feature maps. They won the 1st prize of TuSimple Dataset Challenge.[5] Wang et al proposed LaneNet, which utilizes a group of LSTMs at its output layer. [4]

These state-of-the-art lane detection models also rely on traditional networks like VGG-16 to extract features. Their subsequent layers are usually specifically designed to recognize lanes, thus making their models strong for lane detection. However, the weaknesses are, these specially designed lane detection layers are usually too complex, less intuitive to understand and require a great amount of time to train.

3 Dataset

We use TuSimple dataset [9]. The dataset consists of 2.8K 1-second video clips. Each clip has 20 1280×720 frames and the last (20th) frame has labels in JSON format. Each lane is represented as a list of points in the JSON label file. This dataset is freeway only.

3.1 Data Preprocessing

Each image is paired with a same-sized label image for semantic segmentation. The label image is binary and generated from the JSON file. More concretely, since a lane is a list of points. We first called an OpenCV library function to fit all points for each lane. Then we convert all non-lane pixels to 0 and lane pixels to 1. Next, images are resized to 320×192 to preserve its aspect ratio and avoid random cropping. Finally, 2.8K images are randomly shuffled to train (80%), dev (10%), and test (10%) sets.

4 Methods

4.1 SegNet-based Architecture

SegNet is a proven both time and computational efficient deep convolutional architecture for semantic segmentation. SegNet has an encoder network and a corresponding decoder network, followed by a final pixel-wise classification layer.

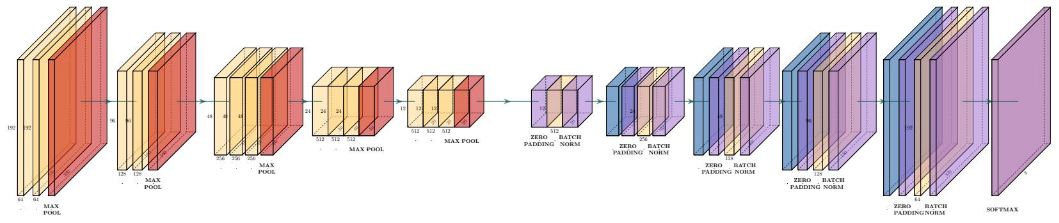

Figure 2: Model Architecture

The encoder network is identical to the VGG16 network designed for object classification. Each encoder layer has a corresponding decoder layer. We reused SegNet architecture in a per-pixel binary classification fashion and optimized loss function. Moreover, instead of having 16 conv layers in the decoder, we simplified SegNet's decoder architecture by reducing the number of decoder conv layers to only 5. (yellow denotes conv layers) The benefit is we have less parameters and the model requires less time to train.

4.2 Loss Function

There are way more non-lane pixels than lane pixels (200 : 1). Intuitively, the model could blindly predict all pixels as non lane but still achieve a very high pixel-wise accuracy. Hence, weighted cross entropy is used to compute the loss function. We penalize the model β times more if the model doesn't predict a lane pixel correctly. β is the weight, and a hyperparameter for tuning.

$$WCE(p, \hat{p}) = -(\beta p log(\hat{p}) + (1-p) log(1-\hat{p}))$$

4.3 Evaluation Metrics

4.3.1 Pixel-wise accuracy

In order to know how accurate is the predicted lane, we calculate the F1 score for the lane class. We will demonstrate why pixel-wise accuracy is not good enough in Section 5.2.1

4.3.2 TuSimple Accuracy

In order to compare the performance to other competitors in TuSimple lane detection challenage, we also calculate the accuracy using TuSimple metrics. After we get the binary prediction matrices, we generate predicated images. In addition to visual comparison, the images are converted back to the JSON labels to align to TuSimple evaluation method. The prediction accuracy is computed as:

$$accuracy = \sum_{clip} c_{clip} / \sum_{clip} s_{clip}$$

where c_{clip} is the number of correct points in the last frame of the clip , s_{clip} is the number of requested points in the last frame of the clip. A point is considered correct if the predicted point and label point is within a certain threshold.

5 Experiments/Results/Discussion

5.1 Hyperparameter Tuning

We use Adadelta optimizer which adapts learning rate dynamically for us. The default initial learning rate and decay factor are used. The architecture is inherited from SegNet, and we don't add or remove layers.

We focused on tuning the most important hyperparameter, class weight β, since it will significantly affect model prediction accuracy. As the pixel ratio (200:1) mentioned above, we search the most proper value from (50, 200, 500, 2000) and plot the training loss and dev loss in the same graph as shown in Fig 3.

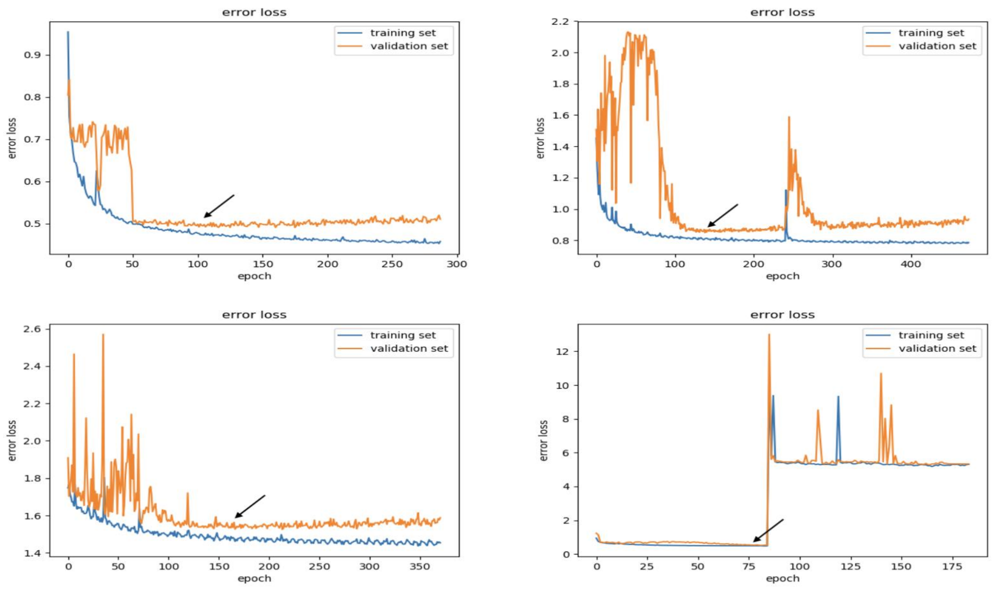

Figure 3: Training loss and Dev loss for different class weight

Here, the gap between training loss and dev loss is getting large which indicates an overfitting. Early stopping technique is leveraged to select the best model for each class weight from the epochs where the black arrows pointing to in the graph. Then the selected best models are used to decide the best class weight by evaluating the model performance on the testing set.

5.2 Model Evaluation

5.2.1 Pixel-wise accuracy

Followings are examples of prediction from the four models using different class weight, the pixel-wise accuracy of test set is listed in the table.

data/weights	50	200	500	2000
train	0.6174	0.6294	0.7318	0.5679
dev	0.6383	0.6242	0.7323	0.5754
test	0.6706	0.6720	0.6245	0.4790

Table 1: F1 score for best model using different class weight

The model using weight 200 has the highest pixel-wise accuracy of 67.2% on the test set and hence 200 is selected as the class weight to do further error analysis and visualizations.

However, it's observed that, although the predictions are quite good, the pixel-wise accuracy is very low compared to the visual result. After a thorough analysis, we think the reasons are from the following aspects:

1. Two types of labeling errors Type I labeling error is as shown in Fig 4 where the ground truth doesn't mark the lane correctly. It is caused by TuSimple data set.

Type II labeling error is as shown in Fig 5 (left) where not all lane pixels are covered. It is caused by labeling method used during converting JSON file provided by TuSimple to images.

2. Information loss during image re-sizing

Lots of re-sizing need to be done since the size of the input images as well as ground truth labels need to be the same as that defined in the first layer of the CNN. The pixel value of ground truth label

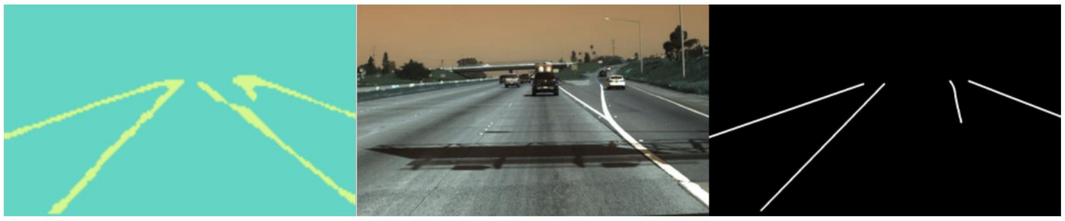

Figure 4: Type I labelling error (Left: prediction image. Center: input image. Right: ground truth)

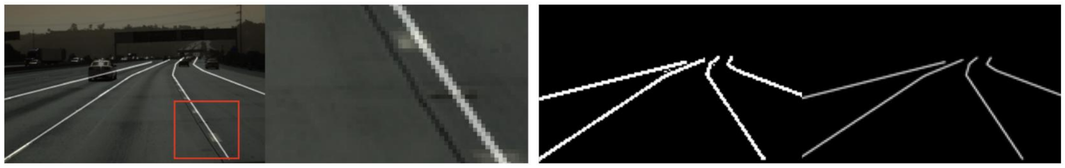

Figure 5: Left part: Type II labeling error. Right part: re-sizing error (left: ground truth image after re-sizing. right: original ground truth image

will be modified by the interpolation algorithm and the recovery threshold we choose as shown in Fig 5 (right).

Hence, we turn to use TuSimple Accuracy as the main metrics since it measures more on the location of the lane instead of the width.

5.2.2 TuSimple Accuracy

We re-evaluate the performance of models using different weights and it turns out that the accuracy of using class weight 200 is still the highest.

We apply LaneNet with provided pre-trained weights to generate predicitons of the test set and compare the accuracy with our model.

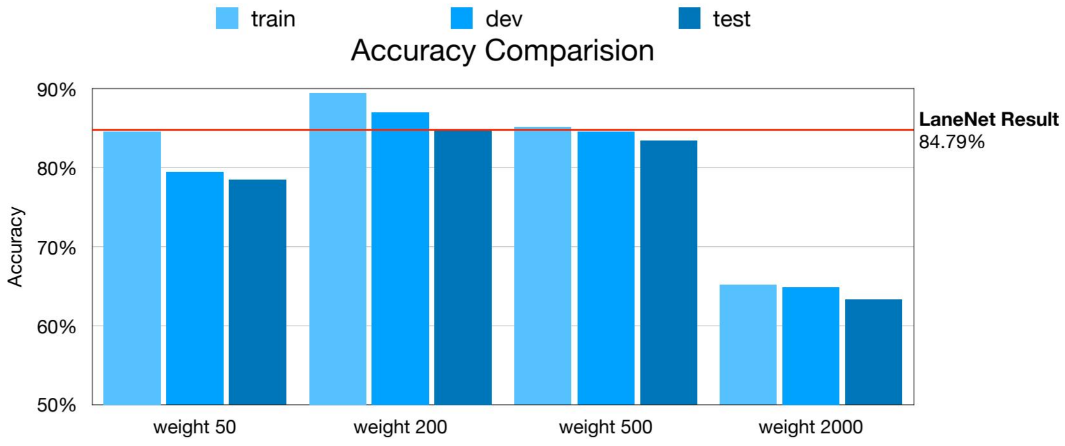

Figure 6: TuSimple accuracy for different class weight (Reference line: TuSimple accuracy of LaneNet on the same testing set

5.3 Result

Finally, we gained the following output of predictions in Fig 7.

In order to visualize the generalization of the model, we find a video and leverage our model to mark the lane frame by frame. Please refer to the video version here: https://github.com/XiangbingJi/Stanford-cs230-final-project

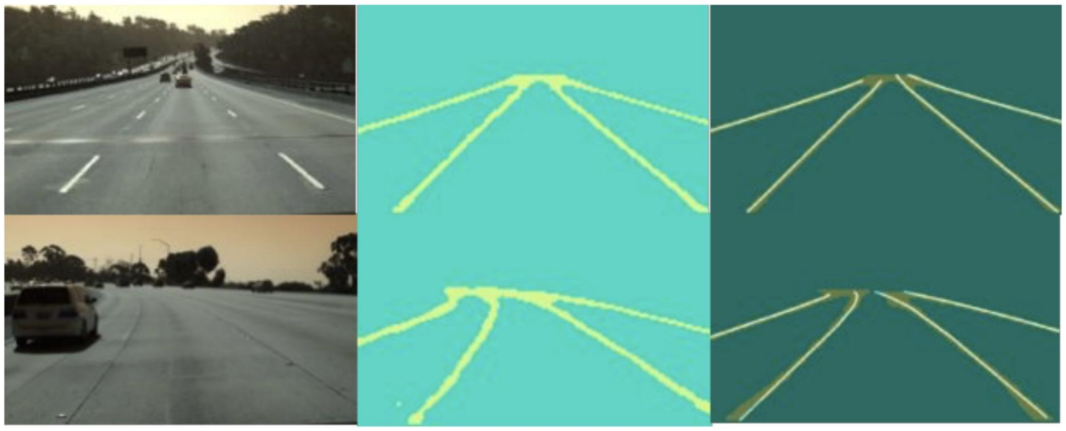
Figure 7: Left Column: input image. Center Column: prediction image. Right Column: Prediction image with overlaid ground truth

6 Conclusion/Future Work

We treated lane detection as a semantics segmentation problem and applied an encoder-decoder CNN architecture to tackle the problem. We achieved an 84.8% accuracy using TuSimple metrics, which is comparable to some state-of-art models like LaneNet. (4th prize in TuSimple challenge).

In the future, an optimized loss function with dynamic class weights[4] can be used to increase prediction accuracy. Collecting more data with different scene types (different weather condition, night, etc.) would likely improve the prediction ability and make the model more robust. Leveraging more advanced regularization techniques like Dropout may help us reduce the overfitting. In addition, using more proper interpolation technique when resizing the images might also help, since we do see some loss when resizing the images. Future works should also include freezing the VGG layers to speed up training, then we are able to iterate faster and try more ideas.

7 Contributions

Junjie worked on dataset searching, dataset pre-processing, data and model visualization. Xiangbing worked on dataset searching, TuSimple Dataset pre-processing, TuSimple Accuracy Generating, LaneNet training and LaneNet result generating (so that we have a model to compare to). Zhengxun modified the original implementation of the SegNet by adding weighted loss function and customized metrics functions, did hyperparameter tuning and training for all class weights, and wrote the scripts for visualizations of loss graphs as well as the video demo.

Three of us worked equally on the model architecture proposing, final poster and final report.

8 Acknowledgement

We would like to thank our project TA Steven Chen for his consistent help and insightful advices.

References

[1] Urmson, Chris, et al. "Autonomous driving in urban environments: Boss and the urban challenge." Journal of Field Robotics 25.8 (2008): 425-466.

[2] Badrinarayanan, et al. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." (2017).

[3] Neven, Davy, et al. "Towards end-to-end lane detection: an instance segmentation approach." 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018.

[4] Wang, Ze, Weiqiang Ren, and Qiang Qiu. "LaneNet: Real-Time Lane Detection Networks for Autonomous Driving." arXiv preprint arXiv:1807.01726 (2018).

[5] Pan, Xingang, et al. "Spatial as deep: Spatial cnn for traffic scene understanding." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[6] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[7] Zhao, Hengshuang, et al. "Pyramid scene parsing network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[8] https://github.com/divamgupta/image-segmentation-keras

[9] TuSimple dataset and explanation: https://github.com/TuSimple/tusimple-benchmark

[10] Github used for plotting model architecture: https://github.com/HarisIqbal88/PlotNeuralNet