CS230

Improvements of SCNN on lane detection in harsh scenarios

Yiyun Gu
guyiyun7@stanford.edu

Yizhi Zhang
zhangyz@stanford.edu

Ziyue Xiao
ziyuex@stanford.edu

Abstract

Spatial CNN is capable of capturing spatial relationships of pixels across rows and columns of images for lane detection through message passing layers. We have explored several methods to improve SCNN performance especially in harsh scenarios in terms of accuracy, F1 and training time. (1)We modify the loss function to make background pixels have lower influence during training. (2)We augment training data by decreasing brightness, adding Gaussain blur and decreasing contrast to existing images. (3)We replace the encoder VGG-16 with ResNet-18 as backbone so that ResNet18 is able to preserve useful low-level information for much deeper layers. (4)We replace Momentum optimizer with SGD+Nesterov momentum optimizer. The first three methods achieve higher training lane accuracy but have lower test F1 score than the baseline model. The last method achieves lower training lane accuracy. With ResNet-18 as backbone, the model trains 4 times faster than the baseline model with VGG16. However, the model with ResNet18 shrinks image size and fails to process sufficient information so the F1 score is much lower.

1 Introduction

Lane detection is vital to a fully understanding of traffic scenes in autonomous driving. In reality, however, detecting lanes can be challenging considering many harsh scenarios, including adverse weather, irrelevant road markers and occlusion by other vehicles. Another challenge in lane detection comes from the long and thin shape of traffic lines, there are fewer pixels than the background pixels, which makes training task even harder.

In our project, we intend to improve the performance of SCNN in more multiple scenarios, like night or occlusion. The input of our model is front-view RGB images of lanes during driving, and the output is the existence value and probmap for four lanes. If the probability of existence is higher than 0.5, we would go over the probmap and produce the the final predictions of lanes.

2 Related work

Traditional models [6] [10] try to predict lanes by generating lane segment information using multiple hand-craft low-level features, like color and edge. These models are simple and can be generalized into various situations, but the performance of such models are greatly affected by external conditions, like lighting condition and occlusions. The cumbersome feature selection process also makes it hard to be popularized [5].

As deep learning take off in computer vision, many deep models are applied to avoid using hand-crafted features and learn to extract features in an end-to-end manner [3]. These approaches usually treat lane detection as a semantic segmentation task, classifying image pixels into different categories and testing if a pixel belongs to a traffic line or not [1].

,

The main problem with these CNN models is that they rely heavily on the segmentation maps of traffic lines, but they can only learn from very subtle and sparse annotation. Increasing the width of lanes may solve the problem to some extent, but it requires more work on pre-processing and may hurt the test performance [3]. Lee et al. [7] used vanishing points as additional supervisory signals, thus alleviated the reliance of deep models on sparse annotations.

In our project, we follow the work of Pan et al. [8] which won the first place in CULane Dataset Challenge. They adopted the idea of "message passing" and utilized spatial information to help propagate information between neurons. But this model doesn't perform very well in harsh scenarios like night and raining days. Therefore, we want to improve the efficiency of this framework in more challenging situations.

3 Dataset and Features

The dataset we use is CULane from Pan et al. [8], a large scale challenging dataset comprised of urban, rural and highway scenes collected in Beijing for traffic lane detection. This dataset extracts 133235 frames from more than 55 hours of videos frames, and the images are divided into 88880 images for training set (66.7%), 9675 images for validation set (7.3%) and 34680 images for test set (26.0%). As shown in Fig.A1 in Appendix, test set is further divided into normal and 8 challenging categories, including crowded, night, no line, shadow, arrow, dazzle light, curve, crossroad. Each frame has a resolution of 1640 x 590 and is distorted. Lane segmentation labels, i.e.per-pixel labels generated from original annotations, are provided.

4 Methods

The baseline model we choose here is the Spatial CNN (SCNN) proposed together with the CULane dataset by Pan et al. [8]. This model utilizes a VGG16 encoder as the feature extractor and then pass the features into the customized layers, which are designed to pass the spatial information in the input features more effectively. Specifically, the customized layers perform convolutions across the dimensions of height and width of the input tensor with 3D kernels, which could be visually represented as Fig.1 below.

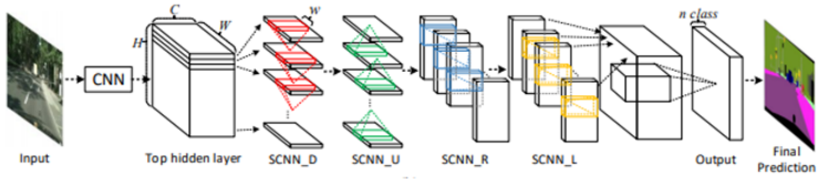


Figure 1: Spatial CNN Implementation[7]

Note that the convolution is carried out in all four directions so as to gain more complete spatial correlations between the features. Mathematically, the "SCNN_D" layer which represents the convolution across the image height from top to down can be written as follows [8]:

$$X'_{i,j,k} = \begin{cases} X_{i,j,k}, & j = 1 \\ X_{i,j,k} + f(\sum_m \sum_n X'_{m,j-1,k+n-1} \times K_{m,i,n}), & j = 2, 3, \ldots, H \end{cases}$$

Where $f(\cdot)$ is the activation function like ReLU, $K_{m,i,n}$ stands for the weight in the kernel from element in the ith channel in the last slice to the element in the jth channel in the current slice with the difference in the width direction being n, and $X'_{i,j,k}$ denotes an updated element at the ith channel, jth row and kth column. The weight values are shared across different slices for one message passing direction so an input tensor with shape (H, W, C) will correspond to a 3D kernel of size (C, C, W).

Intuitively, the SCNN algorithm adds the message passing layers in a way very similar to standard convolutional layers, except that the convolution now happens across the height and width dimensions. Besides, instead of passing the output of convolution to another layer, the algorithm uses the result to produce a new layer, so that the spatial information between elements at different positions of the image can be better propagated, thus yielding a more effective learning.

In this project, we train the baseline model using hyperparameters in the paper [3] and github[2]. The learning rate is 0.01 and the optimizer is Momentum optimizer with momentum equal to 0.9. The train image height is 800 and the train image width is 288. We change the train batch size from 8 to 4 due to the limit of memory and computational cost. The training epoch is set at 5000 since we've found there is no significant improvement after 5000 epochs. The loss function is cross entropy loss, with a combination of existence loss (i.e. whether the model correctly predicts that a lane mark exists) and segmentation loss. The final output of the algorithm will be probability maps that predict the probability of different types of lane marks in the image. We refer the baseline model as **Model_vgg**.

We use pixel-wise accuracy for lane and background, and F1 score as metrics. For any lane, if the model predicts its lane existence and the IoU is larger than 0.3, we consider it as a true positive.

4.1 Loss function and optimizer

The loss function is defined as: loss $= \sum_{i=0}^{4} w_i f(\mathbf{s}_i, \hat{\mathbf{s}_i}) + \beta g(\mathbf{y}, \hat{\mathbf{y}})$. f refers to softmax cross entropy loss function. The binary segmentation loss is a weighted sum of cross entropy loss of lane and background pixels. The coefficient(w_0) of background pixels for the segmentation loss is 0.4 and the coefficients(w_1, w_2, w_3, w_4) of lane marks are 1. g calculates sigmoid cross entropy loss, which represents the loss estimating existence label of each lane. The coefficient(β) of existence loss is 0.1.

Since the number of background pixels is much larger than that of lane pixels, we want to lower their influence in the loss function. We decrease w_0 from 0.4 to 0.2. The model trained using the updated background coefficient is called **Model_bg0.2**. In addition to the modification of loss function, we also try to add Nesterov Momentum the into the optimizer and trained Model_nest.

4.2 Data augmentation

In this project, we intend to improve the accuracy of the model in more complex scenarios like rainy and poor light, while the dataset only contains around 20000 images in such situations. We expanded the dataset from 88880 to 115701 by decreasing brightness, adding Gaussian blur and deceasing contrast to existing images. The proportion of images generated with different method is: decreased brightness (40%), Gaussian blur (30%) and decreased contrast (20%). We refer the trained model based on the augmented training set as **Model_data_aug**.

4.3 Replacing VGG16 with ResNet

Inspired by some recent works [9], we noticed that other than VGG16, ResNet is another commonly used backbone structure for feature extraction, and we want to explore whether ResNet has more advantage over VGG16 when solving this problem. Our hypothesis is that ResNet should outperform the VGG16 network mainly because of its shortcut structure. We believe such structure would make it easier and more efficient for the geometric information to be passed into deeper layers, thus giving more accurate and richer extracted information.

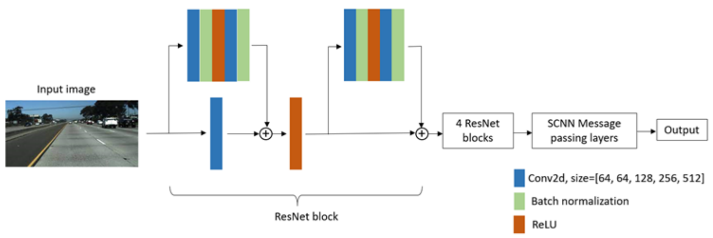


Figure 2: ResNet Backbone System Diagram

For our modified model, the system diagram is shown in Fig.2. The ResNet18 backbone is implemented with 5 ResNet blocks, each containing two layers with a regular convolutional operation path and a shortcut path. The convolution for the first block has a 7×7 kernel to shrink the size of the image so that it will be easier for the model to proceed the information, and the rest four blocks uses 3×3 kernels. The stride is 2 for conv2d and the size of the output from the 5 blocks are in the order of 64, 64, 128, 256, 512. The code implementation of ResNet18 uses a reference from Kim [4] with pretrained weights on ImageNet dataset. The model is then finetuned on our CULane dataset.

5 Results and discussion

The converging trend of training accuracy is shown in Fig.3. Model_bg0.2, Model_data_aug and Model_resnet show better performance and similar performance afterwards after 500 epoch, while the accuracy of Model_nest is much lower than the baseline model. All the three share a lower background coefficient (0.2 or 0.05 for Model_resnet) than the baseline model (0.4), which means background pixels have less influence on loss function. By adding Nesterov momentum, the accuracy of model during training is much lower even though a background coefficient of 0.2 is preserved. We run the test set and get F1 measurement listed in Table 1.

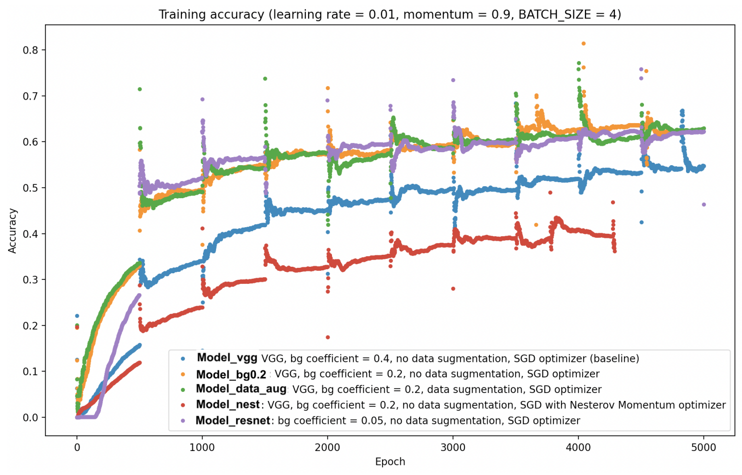


Figure 3: Training Lane Accuracy

5.1 Loss function

Decreasing the coefficient of background pixels in loss function greatly improve training accuracy ((Table A2 in appendix)), but the recall measurement (Table A3 in appendix) is slightly lower that that of baseline model, which results in an overall similar F1 performance with Model_vgg. The training accuracy refers to accuracy in terms of lane marks. With lower background coefficient, the model emphasizes more on lane pixels prediction, thus increasing lane accuracy at the cost of background accuracy. However, true positive depends on Intersection over Union (IoU), which considers the union of predicted and actual lane pixels. Therefore, it's insufficient to simply make the the number overlapped pixels of predicted and actual lane pixels. Decreasing the coefficient of background pixels is not effective.

5.2 Data augmentation

From the F1 measurement, we see that the Model_data_aug doesn't outperform Model_vgg as expected. It has lower accuracy and lower recall in test set. There may be several possible explanations. First we only train the model for 5000 epochs, which is quite limited compared with the epoch number of 90000 in the paper[3]. In addition, the augmented data may not fully represent images in harsh scenarios. For example, blurred images in day light aren't necessarily images at night. Instead, augmented data might add too much noise. It's better to know the category of each image for us to make further data augmentaton.

5.3 ResNet network

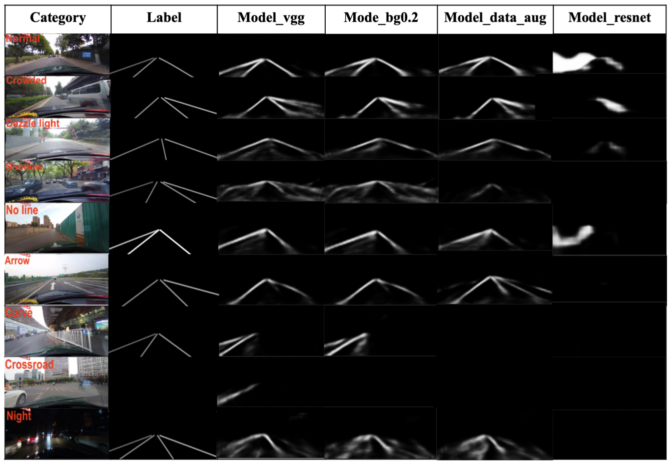
It turns out that the ResNet18 model can achieve a good accuracy compared to other models during the training phase, however it has very low F1 score during test. Upon our observation, one important factor that fails the model in test could be how the input image has been shrunk in the convolutional blocks. The intention of shrinking the size is to have fewer parameters and mitigate the redundant information in the image, and these advantages are observed during training since the runtime of one epoch is only about 20% of the runtime of other models (~ 0.5s v.s. ~ 2.5s). However, the result tells that for the lane prediction problem we do have to seek a high resolution output so that the area of the lane, which is very small compared to the background, could be properly represented. The implementation of VGG16 in [2] doesn't apply max-pooling for the last two convolutional blocks before message passing layers.

Another reason could be that the ResNet structure seems very sensitive to the loss from the background prediction therefore it cannot perform optimization over both the lane and background predictions unless the loss function only takes a minor penalization over the background (background loss coefficient set to 0.05 while others are 1). To address this, one possible solution could be to carefully design a loss function that incorporates more balanced background and lane losses while paying more attention to lane prediction. Further pre-training could also help with this.

Table 1: F1 of different models on testing set (for crossroad, only false positive count is shown)

	Model_vgg	**Mode_bg0.2**	**Model_data_aug**	**Model_resnet**
Normal	67.67%	64.71%	58.48%	16.67%
Crowded	44.12%	40.15%	33.71%	16.51%
Dazzle light	34.06%	32.62%	29.17%	3.30%
Shadow	35.74%	30.74%	21.50%	3.03%
No line	23.94%	25.81%	19.73%	1.79%
Arrow	47.62%	49.92%	37.39%	12.55%
Curve	42.53%	41.40%	30.42%	1.87%
Crossroad	688	389	202	78
Night	38.28%	35.48%	32.69%	2.67%

6 Conclusion



We have implemented several methods that we expected to improve the performance of SCNN but it turns out that the baseline model still achieves the highest F1 score. There is accuracy trade-off between lane and background. By decreasing the coefficient of background pixels in the loss function, the model is more likely to predict pixels as lane and increase lane accuracy. But F1 and IoU considers both lane accuracy and background accuracy. For the data augmentation, we expect blurred images and images with lower bright and contrast to help the model learn better about harsh situations but the augmented data might not represent harsh conditions well and instead add too much noise. Also, the VGG16 backbone of the baseline model doesn't apply the last two max-pooling before message passing. This results that the ResNet18 backbone pass images with smaller size to sequential message layers and the model might not access enough information.

Due to limitation of time, we couldn't explore the ResNet backbone thoroughly to check reasons why it performs much worse. If we had more time, we would like to find a better pretrained ResNet18 and make ResNet18 consistent with VGG16 in terms of dimensions. We will implement other backbones such as ResNet50 and Inception to see whether backbones will boost the performance of SCNN. Also, the training data are mixed among different categories. We would like to find a way to distinguish them so that we could make better data augmentation on harsh scenarios. Moreover, since the original paper trains for 90000 epochs, we would train the model for longer time to check further improvement.

References

[1] Y. Hou. Agnostic lane detection. *CoRR*, abs/1905.03704, 2019. URL `http://arxiv.org/abs/1905.03704`.

[2] Y. Hou. Scnn-tensorflow. `https://github.com/cardwing/Codes-for-Lane-Detection/tree/master/SCNN-Tensorflow`, 2019.

[3] Y. Hou, Z. Ma, C. Liu, and C. C. Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[4] J. Kim. resnet-18-tensorflow. `https://github.com/dalgu90/resnet-18-tensorflow/tree/49eb67c3c57258537c0dcbab5cdf2c38bb1af776`, 2018.

[5] Y. Ko, Y. Lee, S. Azam, F. Munir, M. Jeon, and W. Pedrycz. Key points estimation and point instance segmentation approach for lane detection. 2020.

[6] C. Lee and J. Moon. Robust lane detection and tracking for real-time applications. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):4043–4048, 2018. doi: 10.1109/TITS.2018.2791572.

[7] S. Lee, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T. Lee, H. S. Hong, S. Han, and I. S. Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. *CoRR*, abs/1710.06288, 2017. URL `http://arxiv.org/abs/1710.06288`.

[8] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial as deep: Spatial CNN for traffic scene understanding. *CoRR*, abs/1712.06080, 2017. URL `http://arxiv.org/abs/1712.06080`.

[9] L. Tabelini, R. Berriel, T. M. Paixão, C. Badue, A. F. De Souza, and T. Olivera-Santos. Keep your eyes on the lane: Attention-guided lane detection. *arXiv preprint arXiv:2010.12035*, 2020.

[10] Yinghua He, Hong Wang, and Bo Zhang. Color-based road detection in urban traffic scenes. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):309–318, 2004. doi: 10.1109/TITS.2004.838221.

7 Contributions

All three members of our team are heavily involved in each part of the project, but certain people focus more in specific areas based on their skills. Yiyun Gu from department of Management Science and Engineering and Yizhi Zhang from department of Mechanical Engineering is more responsible for the application of ResNet network in SCNN, and Ziyue Xiao from Civil and Environmental Engineering take lead on loss function improvement and data augmentation.

8 Appendix

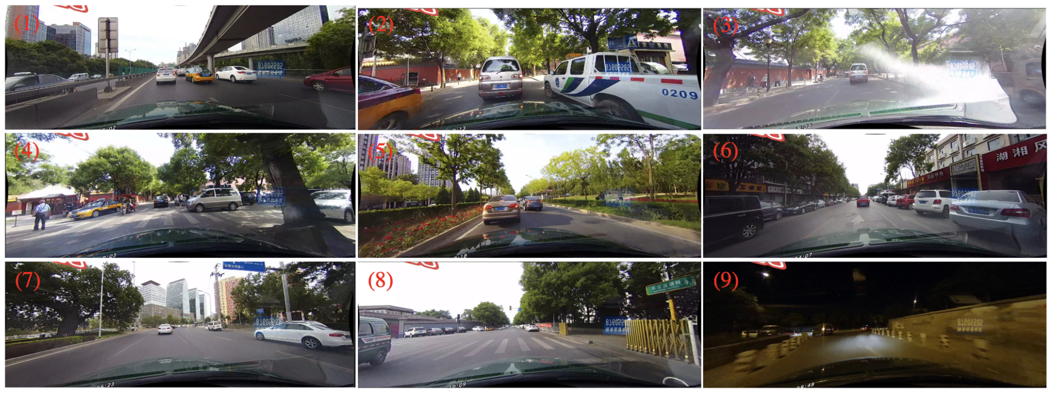


Figure A1: Dataset image examples under different scenarios
(1) normal, (2) crowd, (3) light, (4) shadow, (5) noline, (6) arrow, (7) curve, (8) cross, (9) night

Table A2: Accuracy of different models on test set

	Model_vgg	**Mode_bg0.2**	**Model_data_aug**	**Model_resnet**
Normal	56.46%	66.22%	53.19%	28.54%
Crowd	39.43%	47.75%	37.77%	14.99%
Dazzle light	32.21%	40.26%	30.57%	3.32%
Shadow	33.72%	36.77%	24.88%	8.03%
No line	26.89%	31.79%	22.22%	9.91%
Arrow	43.19%	52.56%	38.43%	17.83%
Curve	37.13%	44.66%	32.04%	7.38%
Cross	N/A	N/A	N/A	N/A
Night	34.09%	36.14%	29.78%	2.44%

Table A3: Recall of different models on test set

	Model_vgg	**Mode_bg0.2**	**Model_data_aug**	**Model_resnet**
Normal	98.59%	96.63%	97.99%	91.76%
Crowd	98.28%	96.70%	97.74%	96.53%
Dazzle light	98.65%	97.23%	98.38%	98.78%
Shadow	98.94%	98.05%	98.94%	97.89%
No line	98.62%	97.43%	98.57%	96.62%
Arrow	98.51%	96.89%	98.18%	95.46%
Curve	98.94%	97.73%	98.71%	85.81%
Cross	99.19%	98.70%	99.25%	99.67%
Night	98.89%	98.04%	98.74%	99.22%