Lane Detection with Deep Learning

Rahil Modi [1], Siddhesh Bagkar [2], Ankit Verma [3]
Department of Automotive Engineering, Clemson University
International Center for Automotive Research (CU-ICAR)
2019 Fall Course Report: AuE 8930 HPC for Autonomy, Instructor: Dr. Bing Li

Abstract

Modern vehicles are incorporated with advanced ADAS systems like lane assist systems which determines whether the vehicle stays within the lanes or not. Also, lane detection is necessary for fully Autonomous Vehicles. Traditional methods are very computationally expensive, and chances of error are very high in them. Most recent methods use deep learning and pixel-wise lane segmentation. In this project, we are using instance segmentation where each lane forms its own instance and it will be trained from end-to-end. We will be applying a learned perspective transformation in addition to the fixed "birds-eye view" transformation. This ensures robust lane fitting even if the road plane changes.

1. Introduction

Perception is one of the most important aspects of an autonomous vehicle. Perception is what tells the vehicle where it is and what is the environment around it and later this helps the vehicle decide its motion how it must proceed. This uses various sensors and modules to determine the environment around the vehicle.
Camera-based lane detection is what is used to detect the lanes and position the vehicle in between the lanes properly and prevent lane departure or something like that. Real-time lane detection is very difficult but it is what is necessary for the fully autonomous vehicle, with that the speed of the lane detection should be very fast because the vehicle would move at considerable speed and if the lane detection is slow than by the time the sensor gives a signal to the vehicle turn, the vehicle will have passed that spot where it had to take a turn and it will cause error and vehicle will move in zigzag way which may cause it to crash eventually. There are many traditional methods to determine the lanes some of them are color-based features [i], the structure tensor [ii], the bar filter [iii], ridge features [iv], etc. The traditional methods rely on highly specialized, handcrafted features and heuristics to identify the lanes and they are combined with Hough transform and particle, Kalman Filters, etc. After identifying the lanes some post-processing is required to eliminate the misdetections and join the segments to form the final lane. The issue with these model-based systems is robustness, with a sudden change in road scenes that can lead to a large error in the detection of lanes.
The traditional methods are replaced with deep networks which makes the model learn dense predictions i.e. pixel-wise segmentation. Some of the examples of these are, in this paper it uses a "*pixel-hierarchy feature descriptor to model contextual information and a boosting algorithm to select relevant contextual features for detecting lane markings*" [v], in this paper, they "*combine a Convolutional Neural Network (CNN) with the RANSAC algorithm to detect lanes starting from edge images. Note that in their method the CNN is mainly used for image enhancement and only if the road scene is complex, e.g. it includes roadside trees, fences, or intersections*." [vi], in this paper it "*introduces the Dual-View CNN (DVCNN) that uses a front-view and a top-view image simultaneously to exclude false detections and remove nonclub-shaped structures respectively*." [vii]. The main disadvantages of these methods are that they can only identify predefined or fixed number of lanes since every lane has a class designated to it and they cannot adjust to lane changes. In this project, we are trying to element these disadvantages and our model is based on the model of this paper [viii]. We will be using instance segmentation to overcome these problems. Where each lane forms its own instance, there is a branched multitask network for lane instance segmentation consisting of lane segmentation branch and lane embedding branch. The lane segmentation has 2 outputs background or lane and the embedding branch disentangles the segmented lane pixel into different lane instances. By splitting we can utilize the full power of the lane segmentation without the need to assign different classes to different lanes. And with the help of the clustering loss function in which it assigns a lane ID to each pixel from lane segmentation without the background pixels. By doing that we can handle variable lanes. In the end, all of them will be converted into the parametric description. And with the help of the polynomial curve fitting algorithms, we will be fitting the lanes onto the image. Most widely used are cubic polynomials, clothoid or splines. To increase the computational efficiency and increase the

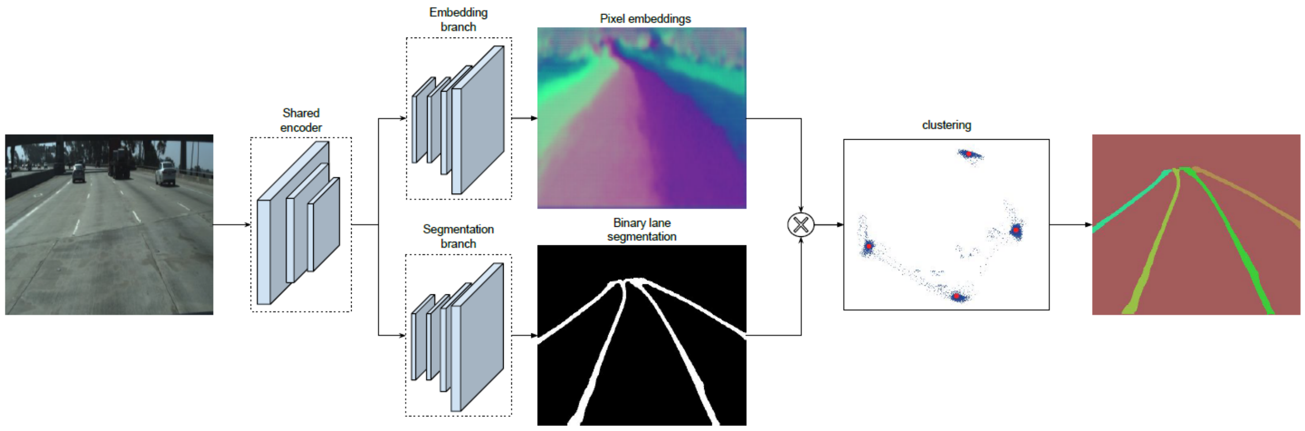

Fig 1 LaneNet architecture. It consists of two branches. The segmentation branch (bottom) is trained to produce a binary lane mask. The embedding branch (top) generates an N-dimensional embedding per lane pixel so that embeddings from the same lane are close together and those from different lanes are far in the manifold. For simplicity, we show a 2-dimensional embedding per pixel, which is visualized both as a color map (all pixels) and as points (only lane pixels) in an XY grid. After masking out the background pixels using the binary segmentation map from the segmentation branch, the lane embeddings (blue dots) are clustered together and assigned to their cluster centers (red dots). [8]

the quality most of the time we convert the image to bird's eye view using transformation and do the curve fitting and getting it projected back onto the image with the help of the inverse transformation matrix. A fixed transformation is not valid when the plane of the road changes. We will be using perspective transformation which is used in this paper which will help us eliminate the problem of change in plane. With this, the lane fitting will be more robust to change of plane.

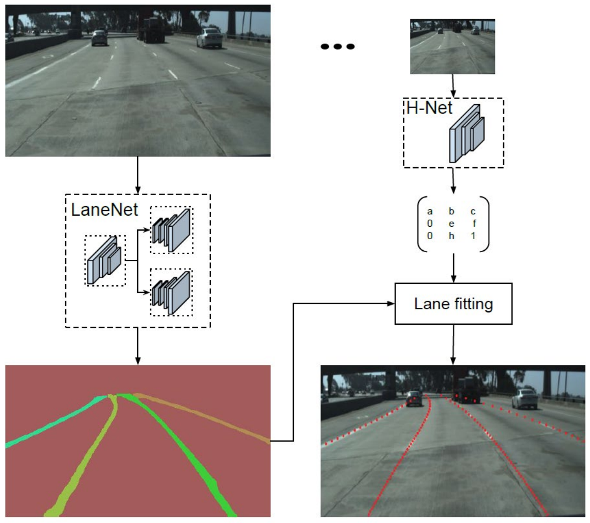

Fig 2 System overview of what is the project about and what are the inputs and the outputs. [8]

2. Methods

We will be training a neural network for an end to end lane detection in this way we will eliminate the problem of model not able to detect lanes when there is a change in a number of lanes and change in the plane. For this we have used instance segmentation approach. Our network combines the advantages of the binary lane segmentation with a clustering loss function designed for one-shot instance segmentation. In the output, each lane pixel is assigned with its own lane id. The output is a collection of pixel lanes where we must fit the curve through those pixels. Ideally, the pixels are first projected into the "bird's-eye view" representation using a fixed transformation matrix. But the fixed transformation matrix has some disadvantages which we have discussed earlier so we are training an H-net that uses ideal perspective transformation. In this, the lane can be optimally fitted with a low order polynomial instead of the traditional "bird's-eye view" transformation.

The instance segmentation task is divided into 2 parts, a segmentation part, and a clustering part.

Binary segmentation: In this, we have trained the model to give a binary segmentation map as an output. In the binary segmentation map, it is trained to show which pixels are of the lane which is not of the lane. To geta ground-truth map we have connected all the points of ground truth lane together forming a single line. We have also decided to the draw lanes over the objects which are blocking the lanes like a car or in absence of a proper lane it will still draw a lane over it, this way we will be training it to predict lane location even when the lanes are obstructed by a car or in improper weather conditions.

Instance segmentation: The second branch of the model is trained with lane instance embedding. The most used detect-and-segment approach is not suitable for the lanes because they use the bounding box detection which is suitable for compact things and lanes are not compact. We are using the one-shot method based on the distance metric learning proposed in this paper [ix]. By using their clustering loss function we are training the instance embedding branch to give an output of embedding for each lane so that the distance between the pixel embeddings of the same lane is smaller and pixel embeddings of different lanes are larger which will lead to clustering of the pixel embedding of the same lanes. This is done with the help of a variance term that applies a pull force to pull each embedding towards the mean and a distance term pushes the cluster centers away from each other.

$$\begin{cases} L_{var} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}[\|\mu_c - x_i\| - \delta_{\text{v}}]_+^2 \\ L_{dist} = \frac{1}{C(C-1)}\sum_{c_A=1}^{C}\sum_{c_B=1, c_A\neq c_B}^{C}[\delta_{\text{d}} - \|\mu_{c_A} - \mu_{c_B}\|]_+^2 \end{cases}$$

Clustering

The clustering is an iterative process. It is done until all the lane embeddings are assigned to a lane. We are using mean shift to make the embeddings shift closer to the cluster center and then do the thresholding

Network architecture

Network architecture is a combination of encoder and decoder network ENet [x]. This has more encoder parameters than the decoder so sharing encoder between the 2 tasks will lead to unsatisfying results. The ENet encoder has 3 stages whereas the network uses 2 stages so the 3rd stage of the encoder and the decoder act as the backbone of each separate branch. Each branch has its loss terms equally weighted and backpropagated through the network.

Curve fitting

First, the lane pixels are transformed into a transformation matrix before fitting of the curve takes place outputted by the H-net. outputted by H-Net. "Given a lane pixel, $\mathbf{p_i}$ = $[x_i, y_i, 1]^T \in \mathbf{P}$, the transformed pixel $\mathbf{p^I i} = [x^I i, yi\text{I}, 1]^T \in \mathbf{P^I}$ is equal to H$\mathbf{p_i}$. Next, the least-squares algorithm is used to fit an n-degree polynomial, $f(y^I)$, through the transformed pixels $\mathbf{P^I}$. To get the x-position, x^*i of the lane at a given y-position y_i, the point $\mathbf{p_i} = [\ , y_i, 1]^T$ is transformed to $\mathbf{p^I i} = \text{H}\mathbf{p_i} = [\ , yi\text{I}, 1]^T$ and evaluated as: $x^I i* = f(yi\text{I})$. Note that the x-value is of no importance and indicated with'-'. By re-projecting this point $\mathbf{p^I i*} = [x^I i*, yi\text{I}, 1]^T$ *into* the original image space we get: $\mathbf{p^* i} = \text{H}^- 1\mathbf{p^I i*}$ with $\mathbf{p^* i} = [x^* i, y_i, 1]^T$. This way, we can evaluate the lane at different y positions." [8]

Loss Function

For getting the output from the H-net in the form of a transformation matrix that is optimal for fitting a polynomial through lane.

Given N ground-truth lane points $\mathbf{p_i} = [x_i, y_i, 1]^T \in \mathbf{P}$, we first transform these points using the output of H-Net:

$$\mathbf{P'} = \mathbf{HP}$$

with $\mathbf{p'_i} = [x'_i, y'_i, 1]^T \in \mathbf{P'}$. Through these projected points, we fit a polynomial $f(y') = \alpha y'^2 + \beta y' + \gamma$ using the least squares closed-form solution:

$$\mathbf{w} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{x'}$$

with $\mathbf{w} = [\alpha, \beta, \gamma]^T$, $\mathbf{x'} = [x'_1, x'_2, ..., x'_N]^T$ and

$$\mathbf{Y} = \begin{bmatrix} y_1'^2 & y'_1 & 1 \\ \vdots & \vdots & \vdots \\ y_N'^2 & y'_N & 1 \end{bmatrix}$$

for the case of a 2nd order polynomial. The fitted polynomial is evaluated at each y'_i location, giving us a $x_i'^*$ prediction.

These predictions are projected back: $\mathbf{p_i^*} = \text{H}^{-1}\mathbf{p_i'^*}$ with $\mathbf{p_i^*} = [x_i^*, y_i, 1]^T$ and $\mathbf{p_i'^*} = [x_i'^*, y'_i, 1]^T$. The loss is:

$$Loss = \frac{1}{N}\sum_{i=1,N}(x_i^* - x_i)^2$$

Since the lane fitting is done by using the closed-form solution of the least squares algorithm, the loss is differentiable. We use automatic differentiation to calculate the gradients.

network architecture The network architecture of H-Net is kept intentionally small and is constructed out of consecutive blocks of 3x3 convolutions, batchnorm and ReLUs. The dimension is decreased using max pooling layers, and in the end 2 fully-connected layers are added.

This way decided the Loss function [8]

3. Results

We got a total fps of around 50 for our model. We have compared our model with an online pre-trained model [7]. We tested around 50 images taken randomly from Google and gave as an input to both the models and analyzed the results.

	Pretrained Model	**Our Model**
Fps	58	50
Success Ratio	48/50	46/50
Dataset used	TuSimple	TuSimple
Training accuracy	98%	91%

Table 1 Comparison between the pre-trained model and our model

Our model was able to detect the change in planes also but it was not able to detect some lanes in some of the pictures and we think the possible error for it dues to polynomial curve fitting not working properly when the image is viewed from the bird's eye view and when it is projected back onto the image that is where the error is occurring. Due to time constraints, we were not able to investigate this error to solve it.

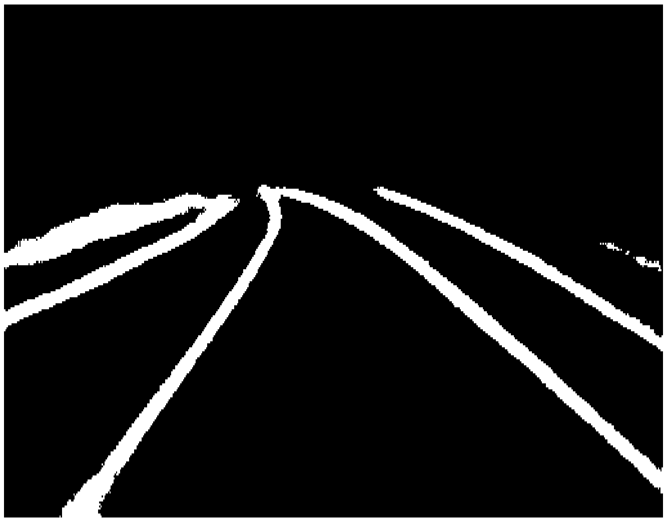
Figure 3 This is the output of a binary lane segmentation [8]

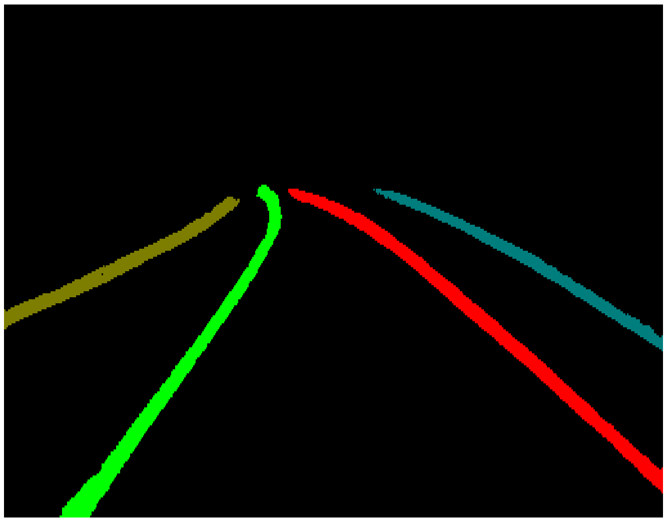
Figure 4 This is the output of the lane instance segmentation [8]

Individual Team Member Contribution:

Rahil: My role was to help Ankit with dataset selection and filtering of it, after that I helped Siddhesh with the training of the model and in the end, I tested the model. We faced many errors, my role was to eliminate those errors and look for another pretrained model, compare between our model and pretrained model. After comparison, I also looked at the analyzation of the results from both the models. Later I tried to analyze why our model was not displaying the lanes properly even though it was able to detect it properly in the binary segmentation. So, there was some problem when it got merged with the instance segmentation and projected back on to the image. Due to time constraints, I could not analyze what was causing the problem. Also, deep learning was very complex for me but I took it as our group project to learn how it is done and I have learned many things from this project and could not learn everything in detail and technical terms of it so I had to take help from online materials.

Siddhesh: My role was to help Rahil with the training of the model and help him write the algorithm. We searched for some papers we could take help to develop our model.

Ankit: My role was to look for a compatible dataset we could use for training our model and filtering of it and later help Rahil & Siddhesh in their work.

4. References

[1] K.-Y. Chiu, S.-F. Lin, Lane detection using color-based segmentation. Intelligent Vehicles Symposium, pp. 706-711, 2005.
[2] H. Loose, U. Franke, C. Stiller, Kalman particle filter for lane recognition on rural roads. Intelligent Vehicles Symposium, pp. 60- 65, 2009.
[3] Z. Teng, J.-H. Kim, D.-J. Kang, Real-time Lane detection by using multiple cues. Control Automation and Systems, pp. 2334-2337, 2010
[4] A. Lopez, J. Serrat, C. Canero, F. Lumbreras, T. Graf, Robust lane ´ markings detection and road geometry computation. International Journal of Automotive Technology, vol. 11, no. 3, pp. 395-407, 2010.
[5] R. Gopalan, T. Hong, M. Shneier, R. Chellappa, A Learning Approach Towards Detection and Tracking of Lane Markings. IEEE Trans. Intelligent Transportation Systems, vol. 13, no. 3, pp. 1088-1098, 2012
[6] J. Kim, M. Lee, Robust Lane Detection Based On Convolutional Neural Network and Random Sample Consensus. ICONIP, pp. 454- 461, 2014.
[7] B. He, R. Ai, Y. Yan, X. Lang, Accurate and robust lane detection based on Dual-View Convolutional Neutral Network. Intelligent Vehicles Symposium, pp. 1041-1046, 2016.
[8] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans and L. V. Gool, "Towards End-to-End Lane Detection: an Instance Segmentation Approach," 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018, pp. 286-291.
doi: 10.1109/IVS.2018.8500547
[9] B. De Brabandere, D. Neven, L. Van Gool. CoRR abs/1708.02551, 2017.

[10] A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, ENet: A deep neural
network architecture for real-time semantic segmentation. CoRR
abs/1606.02147, 2016.