**Lane Detection with Deep Learning**

Project Overview

When a human drives a car, the most common task is to keep the car in the traffic lane. As long as the driver has not been distracted while driving, this task is easy and possible for someone with basic training. On the other hand, for computers, the task of keeping the car between the traffic lines is not as easy as it is for a human.

The reason behind this difficulty is that computers do not naturally have the ability to understand their surrounding environment. The yellow and white lane markings are not understandable for computers inherently. Thank to various kinds of advanced techniques such as computer vision and deep learning, this task has become feasible for computers. This project explores a deep learning method for lane mark detection on streaming videos of road right of way views.

Why this matter?

Fully autonomous driving heavily relies on a proper understanding of the environment around the vehicle. Various perception modules are used for this understanding, and many pattern recognition and computer vision techniques are applied for these perception modules. Lane detection, which identifies the drivable area on a road, is a major perception technique required for autonomous driving.

Problem Statement

It is true that human beings can identify lane markings while driving with basic training, but sometimes, based on the number of crashes and accidents, it is understandable that they can have a disadvantage of not always being attentive. Although, it is not that easy for computers to learn to reliably identify lane line markings but after learning the task, they have the advantage of being consistent and not being distracted by the features and evets that distract humans. In this project, a pre-trained model (PINet[[1]](#footnote-1)) was re-trained to detect traffic lane marks. The model is based on a Convolutional Neural Network (CNN) which performs well on image dataset. CNNs perform well with images by looking at them in a pixel level.

Fundamentals of The Current Model

Deep learning methods show outstanding performance for complex scenes. Among deep learning methods, Convolutional Neural Network (CNN) methods are primarily applied for feature extraction in computer vision.

Lane-mark detection and road region segmentation are the two most popular and common techniques for identifying lanes. The main goal of segmentation is to partition an image into regions. This project uses lane-mark detection method for lane identification.

Semantic segmentation methods, the major research area in computer vision, are frequently applied to traffic line detection problems to make inferences about shapes and locations. Some methods use multiclass approaches to distinguish individual traffic line instances. Therefore, even though these methods can achieve outstanding performance, they can only be applied to scenes that consist of fixed numbers of traffic lines. As a solution to this problem***, instance segmentation*** methods are applied to distinguish individual instances by making clusters.

The method used for this project is based on a combination of the key points estimation and instance segmentation approaches. Key-points are the same as interest points. They are spatial locations, or points in the image that define what is interesting or what stand out in the image.

Pros of Using PINet Model

Most traditional traffic line detection methods extract low-level traffic line features using various handcraft features like color, or edges. These low-level features can be combined using a Hough transform. The combined features generate traffic line segment information. These methods are simple and can be adapted to various environments without significant modification. However, the performance of these methods depends on the conditions of the testing environment such as lighting and occlusion of the dataset. Thus, their reliability as a stand-alone method for lane detection becomes questionable.

The existing methods have certain limitations. The semantic segmentation methods require the labeling or pre-processing at the pixel level for training, which is very time consuming and complicated.

The traditional methods developed based on semantic segmentation generate and predict so many unnecessary points since semantic segmentation useds all pixels for object detection. Semantic segmentation generates plenty of classified pixels even though just a few key points of the lane line is enough to recognize the traffic lines.

In addition, existing methods are not adaptive to various environments according to available computing and processing power. To apply them to light systems like embedded boards, the entire architecture should be modified and trained again.

False negatives, traffic lines that the module fails to detect, do not suddenly change the control values, and correct control values can be predicted from other detected traffic lines or previous results. Some methods have higher rates of false positive that are of more importance due to their higher negative effect on road safety. One of the major advantages of the model used in this project is a very competitive reduction of false positive (the wrong detected traffic lines by module) since this kind of error can cause irrecoverable circumstances.

Image Segmentation Methods

There are two types of segmentation techniques

Semantic segmentation: Semantic segmentation is the process of classifying each pixel belonging to a particular label. It doesn't differentiate across different instances of the same object. For example, if there are 2 cars in an image, semantic segmentation gives same label to all the pixels of both cars. Therefore, this technique would not be practical for this project to identify different lane markings.

A picture containing road, scene, outdoor, street

Description automatically generated

Fig.1. An illustration of a Semantic Segmentation.

Instance segmentation: Instance segmentation differs from semantic segmentation in the sense that it gives a unique label to every instance of a particular object in the image. That is, different lane line markings are assigned different colors and labels (like clusters). With semantic segmentation all the lane marks in an image would have been assigned the same color and label.

An empty road

Description automatically generated

Fig.2. An illustration of an Instance Segmentation.

Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Train** | **Test** | **Resolution** | **Type** |
| Tusimple Dataset | 3,700 | 3,000 | 1280\* 720 | Highway |

Since the dataset (images from Maryland Department of Transportation) initially selected for this project was unlabeled and labeling the images was too cumbersome and time consuming to be done within a couple of weeks,, a labeled public dataset (Tusimple[[2]](#footnote-2)) was selected to re-train the road lane mark detection model. This contains **3626** video clips of 1 sec duration each. Each of these video clips contains 20 frames of which, the last frame is annotated. These videos were captured by mounting the cameras on a vehicle’s dashboard.

This dataset includes X and Y coordinates of at most four ground truth lanes. The annotated lane marks are the two **ego lanes** marks (two lane boundaries in which the vehicle is currently located) and the lanes to the right and left of ego lanes. All the lanes are annotated at an equal interval of height. And all input images are resized to 512 × 256 size and normalized from values of RGB of 0 ∼ 255 to values of 0 ∼ 1 before the data are fed to the proposed network in both training and testing.

Model Training

For lane detection, a neural network has been trained that consists of several hourglass modules.

An Hourglass Module is an image block module mainly and previously used for body pose estimation tasks. The design of the hourglass is motivated by the need to capture information at every scale. The hourglass is a simple, minimal design that has the capacity to capture all of the important features and bring them together to output pixel-wise predictions.

A picture containing engineering drawing

Description automatically generated

Fig.3. An illustration of a single “Hourglass” module. Each box in the figure corresponds to a residual module. The number of features is consistent across the whole hourglass.

Diagram

Description automatically generatedThe network, which has been referred to as the Point Instance Network (PINet), generates points on lane marks and distinguishes predicted points into individual instance.

Fig.5. Architecture of the PINet model which has three main parts. 512\*256 size input data is compressed by resizing network. Then it will be fed to the predicted network which has 4 hourglass modules in this architecture. Each hourglass module has three output branches which predict confidence, offset and embedding feature. The loss function would be calculated from the output of each hourglass block.

The architecture of the model has three main parts.: Resizing network. Predicting network and output branch.

512\* 256 size input data is compressed by the resizing network; the compressed input is fed to the predicting network, which includes four hourglass modules.

An Hourglass Module is an image block module used mainly for key estimation tasks such as pose estimation. The hourglass is a simple, minimal design that has the capacity to capture all of the important features and bring them together to output pixel-wise predictions.

Three output branches are applied at the ends of each hourglass block; they predict confidence, offset, and embedding feature.

Confidence and offset branches predict exact points of traffic lines and embedding feature branch generates the embedding features of each predicted points meaning the clusters, distinguished instances.

The loss function can be calculated from the output of each hourglass block.

The resizing network output is fed to the prediction part which predicts the exact points on the traffic lines and the embedding features for instance segmentation. This network includes hourglass modules, each including an encoder, decoder and three output branches as shown in figure 6. Each colored block in the figure is a bottleneck module which have been described in fig 7.

The loss function fitting has been designed based on the contents proposed by SPGN to discriminate each instance of the predicted traffic lines. (add reference)

Input RGB image size is 512×256; it is fed to the resizing network. This image is compressed to a smaller size (64 × 32) by the sequence of convolution layers in the resizing network; the output of the resizing network is fed to the predicting network.

An arbitrary number of hourglass modules can be included in the predicting network; four hourglass modules are used in this project. All hourglass modules are trained simultaneously by the same loss function. While training, user can choose how many hourglass modules to use according to the computing power, without any additional training.

Chart

Description automatically generated

Fig.6. Details of hourglass block consisting three types of bottle-neck layers: same bottlenecks, down bottlenecks, and up bottlenecks. Output branches are applied at ends of hourglass layers; confidence output is forwarded to the next block.

Diagram, schematic

Description automatically generated

Fig.7. Details of bottleneck. The three kinds of bottleneck have different first layers according to their purpose.

As shown in Figure 6, there are three kinds of bottle neck: same, down and up bottle necks. For more details about the structure of these bottlenecks can be found in [this article](http://arxiv.org/pdf/2001.06604.pdf)[[3]](#footnote-3). Each output branch has a different channel (Confidence: 1, Offset:2 and Embedding: 4) and the corresponding loss function is applied according to the goal of each of them.

Loss Function

For training, four loss function have been applied to each output branch of the hourglass networks. Confidence value determines if the key points of the traffic line exists. Offset value localizes the exact position of the key points predicted previously through the confidence value. And last but not least, the embedding feature is utilized to make clusters by distinguishing key points into individual instances. Therefore, three loss functions are applied to each cell of the output.

* Confidence Loss:

The confidence output branch predicts the confidence value of each cell. If a key point is present in the cell, the confidence value is close to 1 otherwise it is 0.

* Offset Loss:

From the offset branch, PINet predicts the exact location of the key points for each output cell. The output of each cell has a value between 0 and 1.

* Embedding Feature Loss:

The branch is trained to make the embedding feature of each cell closer if the embedding feature are the same in this instance.

Model Implementation

The resizing network reduces the input image size. All input images are resized to 512 × 256 size and normalized from values of RGB of 0 ∼ 255 to values of 0 ∼ 1 before the data are fed to the proposed network in both training and testing. This network includes three convolution layers which have a filter size of 3\*3, stride 2, and padding size 1. Prelu activation and batch normalization are also applied after each convolution layer.

All images have been resized to 512\* 256 and also, they have been normalized from RGB values of 0 ~ 255 to values of 0 ~ 1 before the data are fed to the network. Various data argumentation such as shadowing, adding noise, flipping, translation, rotation and intensity changing also have been applied.

A sign on the side of a road

Description automatically generatedA car driving on a road

Description automatically generatedA car parked on the side of a road

Description automatically generated

A sign on the side of a road

Description automatically generated

A sign on the side of a road

Description automatically generatedA view of the side of a road

Description automatically generatedA view of the side of a road

Description automatically generated

Shadowing

Rotation

Gaussian Noise

Original Image

Translation

Intensity Changing

Rotation

Source code has been written in Pytorch library and Transfer Learning (TL) approach was used to re-train the pre-trained model on Tusimple dataset.

It is a very common approach to use TL while working with big datasets and complicated networks in deep learning. Training CNNs from scratch is quite rare because of sample size limitations. After re-training the model on the labeled images, the model will be tested on a test dataset (video/ images of Maryland roads).

Accuracy is the main evaluation metric of the TuSimple dataset, defined by number of points correctly predicted by the trained module for the given image divided by the number of ground-truth points in the image. The rates False Negative (number of wrongly predicted lanes divided by number of predicted lanes) and False Positive (number of missed lanes divided by number of ground truth lanes) also have been provided.

Result and Conclusion

In this project, the PINet lane detection model has been re-trained with TuSimple dataset. This lane detection method utilizes both key point estimation and point instance segmentation methods. This model achieves a high performance and a low rate of false positives; And as it is known, false positives could cause major accidents and the lowest the rate the more the safety performance of the autonomous vehicle would be. The table below shows the results of this project after approximately 30 epochs?! The following graph also shows that the accuracy of the model increases as the number of training epochs increases.

|  |  |  |
| --- | --- | --- |
| Accuracy | FP | FN |
| 85.10 % | 0.151 | 0.17 |

Chart, line chart

Description automatically generated

The trend below also shows how the training improved within each epoch.

A picture containing cake, toy, train, birthday

Description automatically generatedA picture containing toy, food

Description automatically generatedA picture containing road, scene, outdoor, track

Description automatically generatedA picture containing indoor, road, light, colorful

Description automatically generated

Loss 5.62

Loss 1.22

Loss 3.51

Loss 7.088

A close up of a road

Description automatically generatedA picture containing road, outdoor, track, train

Description automatically generated

Loss 1.03

Loss 1.58

The model has been tested on the unlabeled dataset (Images of Maryland roads) and also the attached link is the video from route MD3 in Anne Arundel County that has been tested with this model.

Future Work

Expanding the model to detect other important objects such as vehicles, pedestrians, signs and traffic signals as they are key to the successful and safe performance of autonomous vehicles. And also, identifying the position of vehicle with regards to the lane marks would be another approach to improve the model.

References:

1. <https://medium.com/analytics-vidhya/detecting-lanes-using-deep-neural-networks-eebf2d9e3603>
2. <https://arxiv.org/abs/2002.06604>

1. https://arxiv.org/abs/2002.06604 [↑](#footnote-ref-1)
2. https://github.com/TuSimple/tusimple-benchmark [↑](#footnote-ref-2)
3. Arxiv.org/pdf/2001.06604.pdf [↑](#footnote-ref-3)