Improved LaneNet for Lane Detection

by

Elikem Buertey

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

© Elikem Buertey 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Lane detection is a critical component in autonomous vehicles and advanced driver assistance systems (ADAS), enabling accurate lane tracking and vehicle positioning. While traditional lane detection methods based on handcrafted features and heuristics have limitations in challenging environments, the adoption of machine learning (ML) techniques has shown promise. However, many existing ML models struggle with detecting a variable number of lanes, making them less effective in complex driving scenarios.

A simple solution involves the use of High Definition (HD) Maps. HD Maps offer comprehensive road information necessary for autonomous driving, but their high cost and inflexibility pose challenges for frequent updates and modifications. This research proposes an innovative approach, the Improved LaneNet (ILaneNet) network, to strike a balance between ML techniques and HD maps. By augmenting input images with a lane parameter namely the number of lanes, we aim to enhance lane detection accuracy without incurring the prohibitive costs of HD maps. ILaneNet seeks to achieve real-time precision in locating and tracking lane markings, even in challenging conditions like inadequate lighting and intricate road layouts.

The objective of this study is to develop a flexible, cost-effective, and robust lane detection system that adapts to diverse driving scenarios. By incorporating pertinent information into the network, we demonstrate improved adaptability and potential advancements in autonomous driving technologies. We also introduce new evaluation metrics namely capacity, lost capacity and unsafe driving measure to assess lane detection techniques more comprehensively. We also propose evaluation of lane detection techniques by using a lane abstraction approach instead of the traditional line abstraction method. Through extensive evaluation and comparisons, we showcase the superiority of ILaneNet over LaneNet in detecting lanes. This research contributes to bridging the gap between ML techniques and HD maps, offering a viable solution for effective and efficient lane detection in autonomous vehicles and ADAS.

Acknowledgements

I would like to extend my heartfelt gratitude to the numerous individuals who have contributed to the successful completion of this thesis. This academic journey has been a challenging and rewarding experience, and I am profoundly appreciative of the support, guidance, and encouragement I have received.

First and foremost, I want to express my deepest appreciation to my thesis advisor, Professor Sagar Naik. Professor Naik's expertise, mentorship, and unwavering commitment to my academic growth have been invaluable. His guidance not only shaped the direction of this thesis but has also inspired me to push my intellectual boundaries. I am truly fortunate to have had the privilege of working under his guidance.

I would also like to thank the faculty members and experts who served on my thesis committee. Their constructive feedback and insightful suggestions have greatly enhanced the quality of this research. Their dedication to academic excellence has been a constant source of inspiration.

Furthermore, I am grateful to my friends and fellow students who provided moral support and insightful discussions throughout this journey. Their camaraderie and shared enthusiasm for learning have made this experience all the more enriching.

I must also extend my gratitude to my family for their unwavering support and encouragement. Their belief in my abilities and their sacrifices have been a driving force behind my academic endeavors.

Last but not least, I want to thank all the authors, researchers, and scholars whose work has been instrumental in shaping the foundation of this thesis. Their groundbreaking contributions have been an essential resource for my research.

To all those mentioned and those who might not have been explicitly named, your contributions, whether big or small, have been pivotal in making this thesis a reality. I am truly grateful for the collective effort and support that have been instrumental in this academic achievement.

Dedication

This thesis is dedicated to you, my beloved parents and granduncle Ernie, as a token of my profound appreciation for your unwavering support. May this accomplishment reflect the indelible mark you have left on my journey.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Road accidents have become a prevalent global cause of fatalities, with factors like irresponsible driving and inadequately designed road structures contributing to the rising number of unfortunate incidents. According to the World Health Organization [61], approximately 1.3 million people lose their lives due to road traffic crashes, and between 20 and 50 million more suffer non-fatal injuries, with many enduring disabilities as a result of their injuries. Astonishingly, 94 % of these accidents are caused by human error, highlighting the potential for significant reduction if human error could be minimized [40].

Addressing this issue, autonomous vehicles and advanced driver assistance systems emerge as possible solutions to decrease human error. The anticipated benefits of autonomous vehicles include crash prevention, reduced travel times, improved fuel efficiency, and parking benefits, with estimated savings of up to $2000 per year per autonomous vehicle and potentially reaching nearly $4000 when considering comprehensive crash costs [16].

The concept of autonomous guided vehicles (AGVs) has already gained significant traction across various industries, where they are used for patient transportation, automated warehouses, hazardous environments, and controlled human transport within carefully designated areas. The system model for autonomous guided vehicles is given in Figure 1.1.

The system model of autonomous vehicles represent a multi-layered framework that shows how the different components work together to make the autonomous vehicles function. Inputs to the autonomous vehicles are given by sensors on the vehicles. Inputs from

cameras and laser scanners are used by the perception model to perceive its environment. GPS, IMU and other on-board sensors assist in localization of the vehicle. The outputs of the localization and perception module are used in planning and vehicle control.

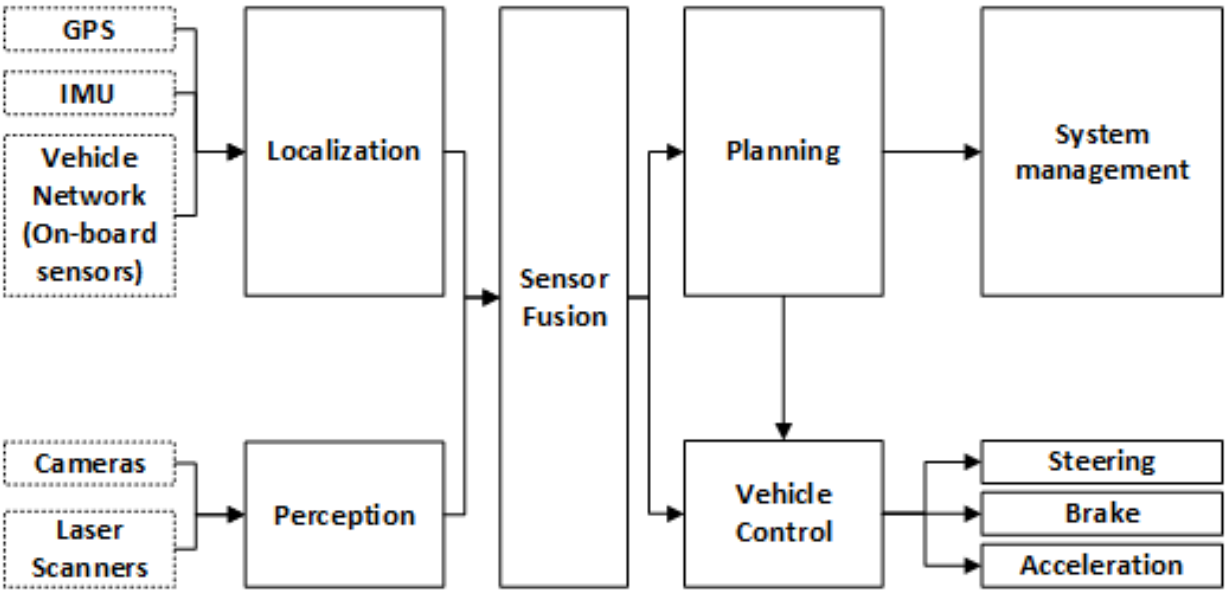


Figure 1.1: Autonomous Vehicle System Model

For further advancements in AGV implementation, certain obstacles must be addressed. One crucial aspect for AGVs' successful navigation in intricate and unstructured environments lies in their vision capabilities. Overcoming challenges related to sensor technology, environmental recognition, and real-time decision-making will be vital to enable AGVs to navigate effectively and safely in complex surroundings.

By leveraging the potential of autonomous vehicles and AGVs, we can aspire to reduce road accidents and enhance overall transportation efficiency. As technology continues to evolve and further research is conducted, the seamless integration of these intelligent systems into our daily lives becomes increasingly promising. This could significantly contribute to safer roads and more streamlined operations across various industries.

1.2 Problem Statement

Lane detection is a crucial vision problem in the context of autonomous vehicles and advanced driver assistance systems (ADAS). It involves the identification and tracking of lane markings on the road to determine the vehicle's position within its lane. Modern cars heavily rely on lane detection to ensure safe and precise navigation on roads and highways. The ultimate aim is to achieve real-time accuracy in locating and tracking the lane markings, even in challenging environmental conditions like inadequate lighting, intense glare, or intricate road layouts.

Several methods have been proposed to detect lanes. Traditional methods which rely on handcrafted features and heuristics were initially proposed . However, this method is prone to fail in challenging scenarios such as adverse weather conditions, occlusions caused by other vehicles, and complex urban road network. In such challenging environments, the limitations of traditional lane detection techniques become evident, emphasizing the need for more robust and adaptable solutions.

With the advent of machine learning and deep learning techniques, researchers have explored their application to the lane detection problem. However, many existing neural networks face limitations in detecting a variable number of lanes, as they are often designed to detect only a fixed number of lanes. By detecting a variable number of lanes, we refer to a scenario in which the lane detection algorithm is capable of identifying and delineating any number of lanes present on the road, without being constrained by a predetermined limit. Although some networks, such as Lanenet, can handle variable lane detection, there is still considerable room for improvement in this area.

One common observation among deep learning networks is that they typically take an image as input and produce an output that represents the detected lanes. However, this approach neglects some valuable publicly available data, such as the actual number of lanes and their widths, which could be beneficial in solving the lane detection problem more accurately.

On the other end of the spectrum, High Definition (HD) Maps have emerged as a potential solution. The authors in study [8] defines an HD map as a map which contains all critical static properties (for example: roads, buildings, traffic lights, and road markings) of the road/environment necessary for autonomous driving, including the object that sensors cannot appropriately detect due to occlusion. However, adopting HD maps for lane detection poses significant challenges. First and foremost the process of generating an HD map is very expensive. Secondly, these maps cannot be easily modified frequently due to the high cost involved in their creation and maintenance.

This thesis aims to find a middle ground between the two prevailing approaches to lane detection. The goal is to augment the input image with other lane parameters while keeping the approach simple, cost-effective, and, most importantly, flexible enough to add or modify relevant information easily.

By exploring this middle ground, we seek to enhance lane detection accuracy by leveraging additional data while avoiding the prohibitive costs and limitations associated with HD maps. Ultimately, we aim to create a robust and adaptable lane detection system that can effectively navigate complex driving scenarios and contribute to the advancement of autonomous driving technologies.

1.3 Objectives

The objectives for this thesis have been categorized into short-term and long-term objectives. The short-term objectives are those that have been achieved within the scope of this thesis. First and foremost, we propose ILanenet as an Improved Lane detection network using the NoL (number of lanes) parameter. By fusing the number of lanes associated with an image into the detection process, ILanenet offers a new approach to address lane detection challenges. In addition, we reinterpret performance evaluation metrics as capacity, lost capacity and unsafe driving measure which offer a more comprehensive and nuanced assessment of lane detection techniques. Furthermore, we introduce a shift in the evaluation paradigm by advocating the use of a lane abstraction approach for assessing detected lanes, diverging from the conventional line abstraction methodology.

The long-term objectives represent broader and more ambitious research goals that extend beyond the current scope of this thesis. These objectives, such as exploring the cost-effectiveness of the augmented lane detection approach compared to High Definition (HD) Maps or dynamically updating the model with new data for improved adaptability, can be pursued in future research by other researchers.

By delineating between short-term and long-term objectives, this thesis has laid the groundwork for further advancements in the field of lane detection, providing a valuable contribution and setting a direction for future investigations to build upon the achieved results and explore new possibilities.

1.3.1 Short-term Objectives

Objective 1: Implement and Understand Lanenet Model

The first objective is to implement an existing lane detection network called Lanenet[41]. By replicating Lanenet, we aim to gain a comprehensive understanding of its architecture, properties, and functionalities. Lanenet requires input images of road scenes or frames captured from a camera mounted on a vehicle, with RGB images of resolution 1280×720 pixels. The Lanenet network will be trained and evaluated on the TuSimple dataset [1], a widely used benchmark dataset for lane detection tasks. The TuSimple dataset[1] contains a large collection of labeled road scene images captured under various driving conditions. For each image, they also provide the 19 previous frames, which are not annotated. The annotations come in a JSON format, indicating the x-position of the lanes at various discretized y-positions. On each image, the current (ego) lanes and left/right lanes are annotated, and this is also expected on the test set. Additionally, in situations involving lane changes, a 5th lane may be added to prevent any ambiguity during the lane detection process. This successful implementation will serve as a reliable baseline for comparing and validating the enhancements proposed in this thesis.

Objective 2: Improve on Lanenet's Detection Capabilities

Building on Objective 1, Lanenet will be modified to incorporate the NoL parameter. This strategic modification improves the accuracy and adaptability of Lanenet. The proposed network, Improved Lanenet, which shall be called ILanenet will build upon Lanenet considered in Objective 1. Instead of feeding only the image into the network as an input, the number of lanes will be fed as input to the network in addition to the input image. The hypothesis is that additional lane parameters will make Improved Lanenet perform better than Lanenet. The number of lanes is chosen as an additional input because the number of lanes on the road is less likely to change and is information that is readily available.

Objective 3: Evaluate the performance of Lanenet and ILanenet

After developing both Lanenet and ILanenet, their performance will be evaluated on the TuSimple dataset[1] named in Objective 1. In pursuit of this objective, both networks will be subjected to testing on the same dataset which encompasses various road conditions and lane configurations. To quantify the effectiveness of the network, metrics namely capacity, lost capacity, accuracy and unsafe driving measure will be used to evaluate the network's

ability to detect and segment lane markings correctly. By conducting this comprehensive evaluation, the thesis aims to demonstrate the superiority and practical applicability of ILanenet over Lanenet.

Objective 4: Visualize and Analyze the Lane Detection Results

Objective 4 of the thesis is to visually analyze the lane detection results, comparing ILanenet with Lanenet. This analysis aims to showcase the effectiveness of incorporating lane numbers alongside the data. By comparing sample images from both approaches, the thesis demonstrates what makes ILanenet outperform Lanenet in accurately detecting and adapting to various lane configurations, thereby proving its improved effectiveness in real-world scenarios.

1.3.2 Long-term objectives:

1. Explore incorporating lane parameters with machine learning approaches that can handle variable lane detection: In this thesis, lane parameters were applied to only one existing start of the art model. But how sure are we that this would generalize well to other existing models? This is a question that can be explored in the future by researchers.

2. Conduct analysis to compare the cost-effectiveness of incorporating lane parameters into deep learning networks with the use of High Definition (HD) Maps: It is expected that our approach of incorporating lane parameters into existing models should have a lower cost compared to HD Maps. This is because incorporating lane parameters takes advantage of already existing data and does not take into account every static object on the road. However, the exact amount of cost that is saved is not known. This can also be an area of research to find out the approximate percentage of savings that result from incorporating lane parameters.

3. Explore methods for dynamically updating the lane detection network with new data, enabling easy and efficient adaptation to changing road conditions: As road conditions change over time, the data used to augment the lane detection network might require updates to reflect these modifications. For instance, additional lanes might be introduced, or road widths could be altered. To address this challenge, future work can explore methods that facilitate rapid and cost-effective incorporation of new data into the lane detection model. By investigating efficient techniques, future researchers

can develop mechanisms to update the model promptly without significant computational overhead or financial burden. This exploration will contribute to enhancing the model's real-world applicability, ensuring it remains adaptable to dynamic road scenarios and leading to further advancements in the field of lane detection.

1.4 Impact of Objectives on Autonomous Vehicles

The objectives outlined in this thesis have a significant impact on the broader picture of lane detection technology and its applications in the field of autonomous vehicles and advanced driver assistance systems.

1. **Improved Lane Detection Accuracy:** Enhancing lane detection accuracy is crucial for the safe and reliable operation of autonomous vehicles. Accurate lane detection ensures that self-driving cars can precisely identify and follow lanes, reducing the risk of accidents and enabling smooth lane changes.

2. **Enhanced Adaptability:** Developing a lane detection network with enhanced adaptability allows autonomous vehicles to navigate diverse and ever-changing road conditions effectively. This adaptability is essential as road layouts and lane configurations can vary significantly, especially in urban environments with complex intersections and construction zones.

3. **Effective Visualization and Analysis:** The visualization and analysis of lane detection results help researchers and developers understand the network's strengths and weaknesses better. This insight can lead to more informed decisions and further improvements in lane detection algorithms.

4. **Identification of Limitations and Enhancements:** Identifying limitations and proposing enhancements guide future research and development efforts. Understanding the scenarios where the network may struggle and finding ways to address these challenges will lead to more robust and reliable lane detection systems.

The successful achievement of these objectives advances the state of lane detection technology, making autonomous vehicles safer, more reliable, and better equipped to handle real-world driving conditions. By contributing to the broader picture of autonomous driving technology, the thesis can pave the way for the widespread adoption and integration of autonomous vehicles into our transportation systems, transforming the way we travel and improving road safety for all.

1.5 Thesis Breakdown

We provide an overview of the structure and content. In Chapter 2, we conduct a comprehensive review of the existing literature concerning lane detection, setting the foundation for our research. Chapter 3 introduces the core of our study, presenting the Lanenet network for Lane Detection, along with its improved version, ILanenet. In Chapter 4, we delve into the experimentation and present the results obtained, offering insights into the performance and efficacy of our proposed ILanenet against Lanenet. Lastly, in Chapter 5, we turn our attention to possible future work, where we discuss potential avenues for further research and development. This early summary serves as a roadmap, providing a glimpse into the forthcoming chapters of this thesis.

Chapter 2

Literature Review

2.1 Lane Detection Methods

In the past two decades, a number of researches have been made in the fields of lane detection and prediction [12, 22, 64, 31]. These approaches can be broadly categorized into two main categories:

1. Traditional Methods
2. Deep-Learning Methods

2.1.1 Traditional Methods

Prior to the emergence of deep learning technology, road lane detection was typically approached through geometric modeling, involving methods such as line detection or line fitting. Fundamental attributes like texture, gradient, geometric configurations, and colors are harnessed to identify and align lane lines in road images. The lane detection procedure involves four main stages: image preprocessing, feature extraction, model fitting, and lane tracking. Image pre-processing techniques consist of conversion of coloured RGB images to grayscale, noise reduction, Region of Interest (ROI) selection, edge detection [13].

ROI selection involves the utilization of three principal methodologies: vanishing point detection, perspective analysis along with a projective model, and sub-sampling [36]. Vanishing point detection has been a common approach adopted by various studies. The idea

behind using the vanishing point is that a correctly estimated vanishing point provides a strong clue about the region to localize. The authors in study [26] tackled a similar problem, road detection, by a two step process: the estimation of the vanishing point associated with the main (straight) part of the road, followed by the segmentation of the corresponding road area based on the detected vanishing point. The subsequent method, referred to as perspective analysis combined with the implementation of a projective model, leverages the concept that parallel lane markings within the real-world plane converge at a vanishing point within the image plane. This approach frequently employs perspective analysis to refine the scope of detection to a precise region, which is then identified as the ROI. Through the skillful establishment of a cohesive projection that interconnects the image plane, real-world plane, and camera plane, the process of extracting the ROI is streamlined. In study [32], a perspective projection model connects the camera and road plane, projecting lane marker edge points onto a road-space grid. The central lane line is defined by points on the grid's upper and lower edges, with each grid segment described by its offset from the lower-left point and the horizontal deviation between endpoints. Sub-sampling constitutes the third strategy employed to ascertain the ROI. In subsampling either a predefined or an adaptive region of the image is used to determine the ROI. Examples are given in [48] [51].

Edge detection operators in image processing can be classified into two fundamental operators: Gradient and Laplacian operators, although there are additional operators that do not strictly adhere to these categories [49]. The gradient method detects edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for the zero crossings in the second derivative of the image to find edges. Gradient based edge detectors include Roberts, Sobel and Prewitt edge detection operators while an example of Laplacian based edge detector is Marrs-Hildreth edge detector. The authors in study [49] studied various edge detectors and concluded that under noisy conditions, Canny, LoG, Sobel, Prewitt, Roberts's exhibited better performance, respectively. They also concluded that Canny's edge detection algorithm has a better performance compared to the others on images.

Generally, there are two kinds of features for extracting lane lines: colors and edges [42]. Lane detection techniques for extracting the lane lines can be grouped into three categories: edge-based methods, color-based methods, and hybrid (edge and color) methods [36]. The Hough transform and its variants, such as, Adaptive Hough Transform and Probablistic Hough Transform are the most popular edge-based methods [36]. Steerable filter is also an edge-based technique that has been applied in many research [47][50][38][56] with good results especially when road markings exhibit a clear and uniformly smooth appearance. Color-based methods have the limitation of being influenced by lighting and are not widely used by researchers, because they are influenced by lighting. An example of a colour based

method, HSILMD, was proposed by the authors in [52]. In HSILMD, full-color images are transformed into HSI color representation within a region of interest (ROI) to detect the road surface on the host vehicle. Using the Fuzzy c-Means algorithm, intensity distribution differences within an ROI row of pixels are recorded and clustered, enabling lane marking detection via selected intensity and saturation thresholds. Hybrid methods usually combine width, length, and location of lines with gray levels and brightness values of pixels, which improve the extraction results. An example is given in [5].

Images captured by vehicle cameras are captured in a continuous sequence. This sequential nature of image acquisition allows for an overlap between lanes detected in the current frame and those from the preceding frame. By leveraging information from both the current and previous frames, we can anticipate lane positions and track their evolution over time, enabling a more robust and accurate lane tracking process. Common trackers include Kalman filters and Particle filters[13].

2.1.2 Deep-learning-based Methods.

Deep-learning algorithms are the future, as they are highly adaptive and self-learning. They increase lane detection and recognition ability to a new level. Convolution neural network (CNN) possess some unique properties such as high detection accuracy, automatic feature learning, and end-to-end recognition. Deep Learning Lane detection methods can be grouped into four main categories according to [67]: Encoder-decoder CNN, FCN with optimization algorithms, CNN+RNN and GAN model.

1) Encoder-decoder CNN: A common application of the encoder-decoder CNN architecture is observed in semantic segmentation tasks [67]. Typical examples include LaneNet[41] and IBN-Net [33]. In the original LaneNet which we would improve upon in this paper, an encoder-decoder network was used for binary and instance segmentation. Binary segmentation consists of segmenting the pixels into lanes and background. Instance segmentation consists of generating embeddings for lane pixels. IBN-Net improves on LaneNet by using an attention-based encoder-decoder network for lane detection. IBN-Net's encoder-decoder network also generates a binary and instance embedding. The difference between IBN-Net and LaneNet in the encoder-decoder network is that the encoder and decoder are connected by a self-attention layer.

2) FCN with optimization algorithms: In FCN with optimization, optimization algorithms, such as, clustering and subsampling are followed to achieve the goal of lane detection, lane-marking identification, and vanishing-point extraction [67]. For instance, in VPGNet[30], the comprehensive understanding of road scenes is achieved by integrating

multiple tasks through a unified deep learning architecture. The architecture employs the concept of a vanishing point to guide the predictions of lane markings and road regions. By leveraging the vanishing point as a guiding factor, VPGNet optimizes the joint prediction of lane information and road layout, resulting in improved accuracy and robustness in handling complex road scenes. Deep learning methods for lane detection involving clustering is dominated by semantic segmentation algorithms. Image pixels are classified by the deep neural network, and the lane line information is extracted by clustering and other post-processing methods. An example of a deep learning method involving clustering is LaneNet.

3) CNN+RNN: CNN + RNN methods for lane detection are based on the idea that Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can synergistically exploit both spatial and sequential information within road scenes. CNNs are proficient at discerning intricate spatial patterns within images, making them well-suited for recognizing lane markings and road structures. Through a series of convolutional and pooling layers, CNNs are adept at extracting progressively abstract features, which are crucial for accurate lane detection. However, the analysis of road scenes necessitates the consideration of not only spatial features but also the temporal evolution of lane configurations. RNNs excel at modeling sequential data by incorporating memory of prior inputs, thus capturing the dynamic nature of lane movements over frames. By fusing the capabilities of RNNs with CNNs, the combined architecture gains the ability to comprehend both instantaneous lane contexts and their evolution through time [67][27].

4) GAN model: Given that certain lane detection methods rely on semantic segmentation, and given that Generative Adversarial Networks (GANs) are equipped to perform semantic segmentation tasks, GANs can also serve a purpose in lane detection applications. A common strategy to leverage GANs for semantic segmentation includes constructing a loss function for the segmentation network (generator) with two key components: the first relates to precise pixel-wise prediction or label fitness (Lfit), while the second involves an adversarial loss term aimed at maintaining higher-level consistency attributes. These terms are conditioned on the input image, effectively guiding the network's learning process. The role of the discriminator is to evaluate the authenticity and quality of the predictions generated by the generator in a GAN. Examples of this approach are given by [35][19][65].

Deep learning based lane detection algorithms can also be grouped into three types [54]:

1. Semantic segmentation
2. Instance segmentation

3. End-to-end approach

Semantic segmentation

In semantic segmentation, the task is to assign a specific class label to every individual pixel within an image, effectively dividing the image into various meaningful regions or objects. As illustrated in Figure 2.1, this technique is often used in lane detection, where each pixel in the output image is categorized as either belonging to a lane line or representing the background. This fine-grained pixel-level classification provides a comprehensive understanding of the visual content within an image. By segmenting an image into semantically meaningful regions, it becomes possible to extract critical information about the objects and their spatial distribution, making it a crucial component in lane detection.

Figure 2.1: Semantic Segmentation

Instance segmentation

In instance segmentation, the goal is to precisely identify and distinguish individual instances of objects within an image. In the context of the image presented in Figure 2.2, instance segmentation enables the differentiation and labeling of each individual lane line. These lane lines are not only detected but are also assigned distinct and discernible colors or labels, making it possible to track and identify each lane line separately.

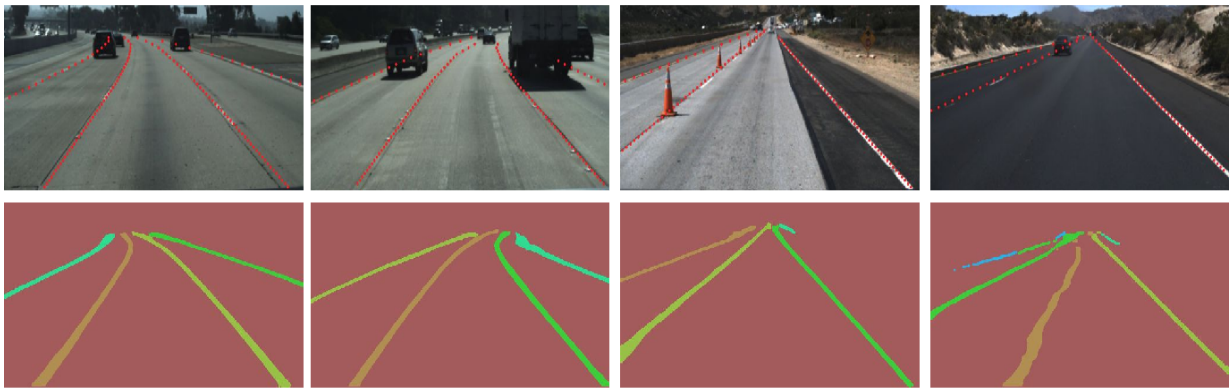


Figure 2.2: Instance Segmentation

End-to-end approach

The goal of lane detection is to find lane lines and estimate the lane lines with a polynomial. In both semantic and instance segmentation, the output of the deep learning pipeline needs to be further processed to detect which polynomial best fits the detected lane lines. However, in the end-to-end approach, the deep learning pipeline outputs an estimate of the polynomial which best fits the detected lane lines and hence requires no further processing.

2.1.3 Datasets

Datasets play a pivotal role in the development, testing, and validation of lane detection algorithms. They contain a wealth of annotated images or videos, often captured by vehicle-mounted cameras or sensors, and are used to train lane detection algorithms. Some common datasets include: TUSimple dataset, CULane dataset, Unsupervised LLAMAS and BDD100K dataset.

1) TUSimple dataset - The TuSimple dataset comprises of 6,408 images captured from US highways, with 3,626 designated for training, 2,782 for testing, and 358 for validation. The test set includes road images captured on US highways, showcasing various weather conditions.

2) CULANE dataset - The CULane dataset consists of a training set of 88,880 samples, a validation set comprising of 9,675 samples, and a test set consisting of 34,680 samples. These datasets were collected from real-world scenarios using cameras positioned on six vehicles driven by different drivers in Beijing. The test set is categorized into several distinct

scenarios, each presenting unique challenges. These categories encompass normal scenarios, crowded scenes, low-light or night scenarios, scenarios with no visible lane markings, shadowy conditions, scenarios with arrow markings, scenarios affected by intense dazzle light, curved road scenarios, and crossroad scenarios. This categorization allows for evaluation of lane detection algorithms on a wide range of real-world driving conditions.

3) Unsupervised LLAMAS dataset - Comprising of 100,042 labeled lane marker images, the unsupervised LLAMAS dataset stems from approximately 350 kilometers of recorded drives. The image labels are automatically generated, first by projecting markers into camera images, and then through further optimization to enhance label accuracy. The dataset annotations include pixel-level annotations for dashed lane markers, as well as the 3D and image space endpoints for individual markers, along with lane associations for each marker. The challenges presented within this dataset encompass a pixel-level binary segmentation problem, a segmentation problem intertwined with lane association, and a lane estimation task.

4) BDD100K dataset - This dataset is drawn from more than 50,000 rides across New York and the San Francisco Bay Area city from streets, residential areas, and highways. It contains 100K driving videos, each lasting 40 seconds. The videos are split into training (70K), validation (10K) and testing (20K) sets. The dataset is made of 720p high resolution images, with a frame rate of 30 fps and GPS/IMU recordings to preserve the driving trajectories. Ten tasks are associated with the dataset: image tagging, lane detection, drivable area segmentation, road object detection, semantic segmentation, instance segmentation, multi-object detection tracking, multi-object segmentation tracking, domain adaptation, and imitation learning.

2.1.4 Integration of Lane Detection Methods

The enhancement of the lane detection system hinges on the amalgamation of various methodologies across algorithmic, systemic, and sensorial levels within the detection framework. Prominent players in the automation sector, including Tesla and Mobileye, have adopted distinctive integration approaches to amplify the capabilities of their lane detection systems [62]. Integration at the algorithmic level entails the fusion of different lane detection algorithms to ascertain accurate lane positions and elevate overall efficiency. In terms of system-level integration, diverse object detection systems collaborate concurrently to enhance real-time communication. Sensor level integration finally complements each other's modality disadvantages and makes the system more robust . An illustrative case of algorithmic integration is evident in the successive fusion of the Hough transform,

RANSAC, and spline model, as exemplified in [60][59]. This fusion adeptly leads to precise lane detection, even in intricate scenarios like steep curves and varying lighting conditions.

In the context of system integration, ongoing investigations revolve around the fusion of lane detection and road detection systems [22][7]. The sequence starts by initially detecting the road area, preceding the lane detection process. This approach serves to expedite lane marking recognition and ensures the precise delineation of the Region of Interest (ROI). The interconnected nature of lane and road boundaries enriches accuracy. The pinnacle of integration lies in sensor level integration—a pivotal phase that combines the distinctive characteristics of each sensor's modalities to enhance the collective functionality of the overarching sensor system, as detailed in [13].

2.2 Lane Detection Evaluation Methods

Evaluating lane detection systems is a crucial step in assessing their performance and ensuring their reliability for real-world applications. Before deploying a lane detection system in real-world situations, it's common practice to evaluate its performance using recorded data or simulated environments. Evaluation allows researchers and developers to assess the system's behavior under controlled conditions, compare its outputs against ground truth data, and identify areas where improvements are needed. Since this evaluation doesn't occur in real time, it provides a comprehensive understanding of the system's strengths and limitations without the constraints of immediate responsiveness.

Some methods have been proposed for evaluating lane detection systems [44][41]. Most Lane detection datasets come with their own metric. For example, CULANE dataset [44] evaluates lane predictions by calculating F1 score, precision and recall as shown in Equations 2.1, 2.2 and 2.3. A lane marking is described as a line having a width equivalent to a certain number of pixels. A lane marking is successfully detected if the Intersection over Union(IOU) of the ground truth and the predicted lane is greater than some threshold. That is, predictions whose IoUs are larger than certain threshold are viewed as true positives (TP).

$$\text{F-measure} = \frac{(1+\beta^2)\cdot \text{Precision}\cdot \text{Recall}}{\beta^2\cdot \text{Precision} + \text{Recall}} \tag{2.1}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.2}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.3}$$

In Lanenet [41], the accuracy, the average correct number of points per image, of lane detection is evaluated using the expression in Equation 2.4.

$$Accuracy = \sum_{im} \frac{C_{im}}{S_{im}} \tag{2.4}$$

where im is an image in the dataset, C_{im} is the number of correctly predicted points in im and S_{im} the number of ground-truth points in im. A point is correct when the difference between a ground-truth and predicted point is less than a certain threshold. The expression for calculating the false positive score and false negative score are given in Equation 2.5 and 2.6 respectively:

$$False\ Positive\ Score\ (FPS^l) = \frac{F^l_{pred}}{N^l_{pred}} \tag{2.5}$$

$$False\ Negative\ Score\ (FNS^l) = \frac{M^l_{pred}}{N^l_{gt}} \tag{2.6}$$

with F^l_{pred} being the total number of falsely predicted lane lines, N^l_{pred} the total number of correctly predicted lane lines, M^l_{pred} the total number of missed ground-truth lane lines and N^l_{gt} being the total number of all ground-truth lane lines.

Chapter 3

Methodology

3.1 Introduction

Traditional lane detection methods typically use a combination of carefully designed hand-crafted features and heuristics to identify lanes in a road scene. These methods often require computationally intensive post-processing and are not easily scalable to different road conditions. In contrast, newer approaches employ deep learning models that are trained to segment lane pixels, even in the absence of visible lane markings. These methods have the advantage of being able to handle a wide range of road scenes. However, they are typically designed to detect a fixed number of lanes, such as the lanes directly in front of the vehicle, ego-lanes, and struggle with detecting lane changes or variable lane configurations.

Among the plethora of deep learning networks used for lane detection, Lanenet was chosen for several compelling reasons. Chief among these is its lightweight architecture, which renders it exceptionally amenable to training and integration within our framework. This deliberate selection of Lanenet stems from its distinct advantages over comparable networks like CLRNET[66] and CULANE[44].

One notable feature of Lanenet is its modest batch size prerequisite. Unlike other networks, such as CLRNET, which mandates a batch size of 32, Lanenet operates efficiently with a notably smaller requirement of just 8. This efficiency not only expedites the training process but also mitigates resource demands, ultimately enhancing overall training efficacy and expediency.

Additionally, Lanenet's innovative two-step approach to lane detection is instrumental in reducing computational overhead. This methodical process effectively minimizes the

number of pixels that need to be clustered by clustering only lane pixels, resulting in a more streamlined and less resource-intensive operation. By design, this approach not only simplifies the network's computational demands but also bolsters its capacity for real-time processing. These factors collectively culminate in a network that not only ensures efficient training but also optimizes computational resources, aligning seamlessly with our project objectives and performance benchmarks.

The Lanenet Network deals with the challenge of identifying variable lane configurations. In the original paper, the problem of identifying lanes is formulated as an instance segmentation problem in which each lane is an instance. The identified lanes are then fitted in bird's eye view before being projected back onto the original image by a perspective matrix. One novelty introduced was predicting the perspective matrix using a neural network. Traditional approaches often relied on a fixed perspective matrix. This resulted in incorrect results when there were changes in the road-plane.

Within this section, we delve into our methodology, starting with the presentation of the well-established Lanenet network. Subsequently, we introduce an enhanced version of the network which we call ILanenet. ILanenet capitalizes on the utilization of not only the input image but also an additional input parameter: the number of lanes depicted (NoL) within the image. This innovative integration significantly enhances the performance of Lanenet.

3.2 Lanenet Architecture

LaneNet is structured as a two-step lane detection network as illustrated in Figure 3.1. Two-step lane detection methods are composed of a feature extracting step and a post-processing step [53].

LaneNet's feature extraction stage comprises of the use of deep learning techniques to semantically segment the image into two distinct categories: binary segmentation and instance segmentation. Binary segmentation classifies the pixels into either a background or lane. Instance segmentation segments the image pixels in such a way as to distinguish lane pixels from each other. This means that not only are the pixels classified as either background or lane, but individual lane pixels are embedded in such a way that pixels that belong to the same lane are similar while those that belong to different lanes are dissimilar.

The post-processing phase focuses on refining the extracted information. This phase primarily involves clustering, which groups lane pixels into clusters. Lane pixels belonging to the same lane will be in the same cluster. Finally, the fitting operation employs

mathematical models to precisely define the trajectory of each lane, further enhancing the accuracy of lane boundary representation. The details of each part of Lanenet's architecture are explained below. Similar processes in both ILanenet and Lanenet have the same number in Figure 3.1 and Figure 3.6.

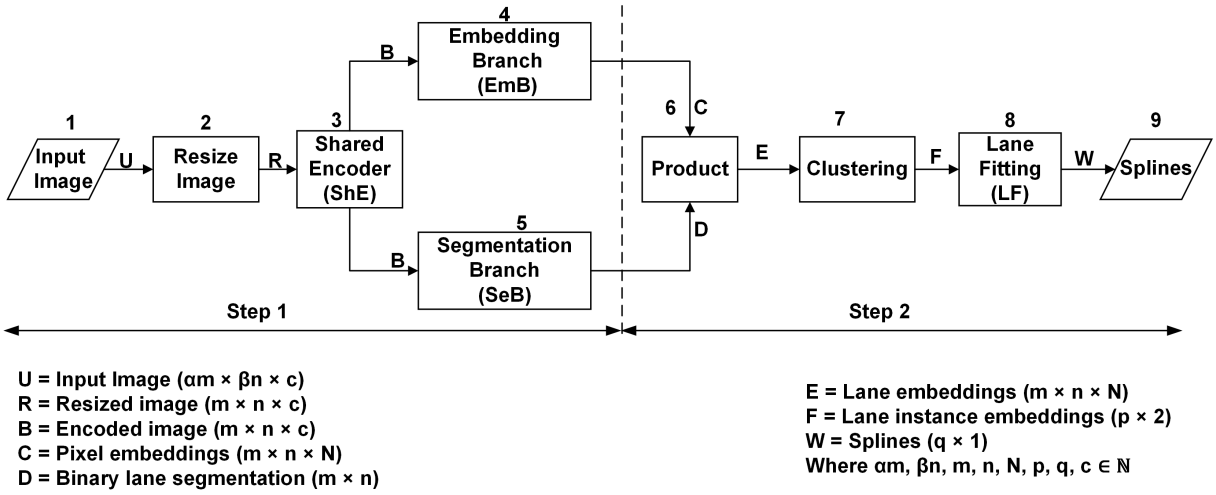


Figure 3.1: Lanenet Architecture [41]

3.2.1 Explanation of Blocks in Lanenet Architecture

Input Image (input 1) and Resize Image (process 2): In the image processing pipeline, input images are resized from their original resolution of $\alpha m \times \beta n \times c$ pixels to a reduced resolution of $(m \times n \times c)$, where $\alpha m, \beta n, m, n, c \in \mathbb{N}$. One of the reasons for resizing the image before feeding it into the network is to reduce the number of parameters and prevent potential memory constraints. When working with neural networks, the size of the input data directly impacts the number of parameters that need to be processed and stored during training and inference. By resizing the image, we effectively reduce its spatial dimensions, resulting in a smaller input size. This reduction in size directly translates to a decrease in the number of parameters required by the network, ultimately helping to alleviate memory-related issues, such as RAM crashes. Resizing the image also provides computational benefits. Smaller input sizes lead to faster processing times, allowing the network to analyze and make predictions more efficiently. Additionally, the reduced parameter count can enhance the overall training speed and optimize resource utilization. However, it's important to strike a balance when resizing the image. If the image is resized

too drastically, crucial details may be lost, potentially compromising the network's ability to accurately detect and interpret features in the image. Hence, it's necessary to consider the trade-off between parameter reduction and maintaining sufficient image resolution for effective analysis and prediction.

Shared Encoder (process 3): The ENet[45] Encoder-Decoder segmentation network was selected due to its efficiency, as it is a fast and compact architecture. This network comprises five stages in total. The first stage contains a single block. Stage 1 consists of five bottleneck blocks. Stage 2 and stage 3 share a similar structure, but stage 3 doesn't downsample the input initially. These first three stages together form the encoder portion of the network. Stages 4 and 5 form the decoder.

Lanenet's architecture is based on the ENet encoder-decoder network, with two notable modifications. Firstly, the output of ENet was adapted to create a two-branched network, accommodating both binary segmentation and instance segmentation branches within Lanenet. Secondly, ENet's encoder had more parameters than its decoder, which results in suboptimal outcomes when the entire encoder was shared[41]. Consequently, in Lanenet, only the first two stages (stages 1 and 2) of ENet's encoder are shared between the two branches, while the full ENet decoder (stages 4 and 5) serves as the backbone for each separate branch. This means that stage 3 of ENet's encoder is not utilized in Lanenet.

In terms of the output, the final layer of the segmentation branch produces a one-channel image for binary segmentation, whereas the last layer of the embedding branch generates an N-channel image, with N representing the embedding dimension. The shared encoder algorithm is given in Table 1 in the appendix section utilizing Algorithms 3, 4, 5, 6 and 7 which are also in the appendix section.

Segmentation branch (process 5): The segmentation branch of lanenet plays a critical role in binary image segmentation, classifying each pixel in an input image as either lane or background. Its primary objective is to identify and delineate lane markings within the road scene, a crucial task for autonomous vehicles and other computer vision applications. However, a common challenge in binary segmentation tasks like lane detection is class imbalance, where the abundance of background pixels greatly outweighs the lane pixels in an image. This imbalance can lead to a skewed learning process, where the model may prioritize classifying most pixels as background, which is suboptimal for accurate lane detection. To combat this, LaneNet addresses the class imbalance issue by employing class-weighted cross-entropy loss during training. By assigning different weights to the two classes, the network is encouraged to pay more attention to the underrepresented lane class, ensuring that it doesn't overlook the critical task of identifying lane markings amidst the sea of background pixels.

The pseudocode for this branch is presented in Algorithm 8 in the appendix section.

Embedding Branch (process 4): The embedding branch of the network is designed to produce embeddings of lane pixels. The embeddings are designed such that lane pixels belonging to the same lane have similar embeddings while lane pixels belonging to different lanes have different embeddings. As outlined in the original paper[41], this is achieved using a clustering loss function which minimizes the distance between pixel embeddings belonging to the same lane while maximizing pixel embeddings belonging to different lanes.

The consequence is that pixels with similar embeddings – pixels that belong to the same lane – will cluster together, forming unique clusters per lane. To do this, a variance term (L_{var}), that applies a pull force on each embedding towards the embedding of a lane is introduced. In this context, the embedding of a lane refers to the mean embedding of pixels belonging to that lane. Also, a distance term (L_{dist}), that pushes the cluster centers away from each other is introduced. Both terms are hinged: the pull force activates when an embedding is at a distance of more than δ_v from its cluster center. The force of pushing between the centers comes into effect only when the centers are at a distance less than δ_d from each other. In this context, δ_v represents the maximum allowable distance between an embedding and the mean embedding of its corresponding cluster. Also, δ_d represents the minimum distance allowed between cluster centers. Let K denote the number of clusters (lanes), N_k the number of elements in cluster k where $1 \leq k \leq K$, x_i a pixel embedding, μ_k the embedding of cluster k, $\|\cdot\|$ the L2 distance, and $[x]_+ = \max(0, x)$ the hinge, the total loss L is equal to $L_{var} + L_{dist}$. The quantities L_{var} and L_{dist} are defined in Equation (3.1).

$$\begin{cases} L_{var} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k} [\|\mu_k - x_i\| - \delta_v]_+^2 \\ \\ L_{dist} = \frac{1}{K(K-1)} \sum_{k_A=1}^{K} \sum_{k_B=1, k_A \neq k_B}^{K} [\delta_d - \|\mu_{k_A} - \mu_{k_B}\|]_+^2 \end{cases} \tag{3.1}$$

The LaneNet clustering process is performed iteratively with the loss function in Equation (3.1) until the network converges. Upon convergence of the network, clusters will emerge in the embeddings of lane pixels. These clusters will exhibit a separation distance larger than δ_d from adjacent clusters, with each cluster possessing a radius smaller than δ_v. The pseudocode for the embedding branch is given in Algorithm 9 in the appendix section.

Product (process 6) and Clustering (process 7): In the binary segmentation map of the image, the background pixels are 0s and the lane pixels are 1s. To isolate the embeddings specific to the lane pixels, the results from the embedding branch and the segmentation branch are multiplied together. This process filters out all non-lane pixels,

leaving us with only the embeddings associated with the lane pixels. Subsequently, we can employ clustering techniques to determine which lane pixels correspond to distinct lane lines. The LaneNet clustering process is performed iteratively, and it involves a specific condition set in the loss function defined in Equation (3.1). This condition requires that δ_d and δ_v satisfy the relationship $\delta_d > 6\delta_v$.

What this condition accomplishes is the ability to select a random lane embedding and subsequently identify other lane embeddings that belong to the same lane. This is achieved by applying a threshold within a radius of $2\delta_v$. This process is repeated iteratively until all lane embeddings have been correctly assigned to their respective lanes.

To ensure that outlier lane embeddings are not inadvertently selected during this thresholding process, a two-step approach is employed. First, a clustering algorithm known as mean shift is used to shift the selected point closer to the cluster center. This step helps to refine the initial selection. After this shift, the thresholding process is applied, which results in the accurate identification of lane embeddings within the specified radius, ultimately preventing the inclusion of outliers.

Lane Fitting (process 8) and Splines (output 9): The original LaneNet implementation introduced H-Net for lane fitting. H-Net was a neural network that predicted the perspective transform matrix which will be used to convert the lane to Bird's Eye View (BEV). The working of H-Net is described as follows:

Let a lane pixel $\mathbf{p_i} = [x_i, y_i, 1]^T \in \mathbf{P}$, where $\mathbf{P}$ is the set of pixels belonging to a particular lane. The transformed pixel to BEV is given by $\mathbf{p'_i} = [x'_i, y'_i, 1]^T \in \mathbf{P'}$ is equal to $\mathrm{H}\mathbf{p_i}$ where H is the output of H-Net. Next, the least-squares algorithm is used to fit an n-degree polynomial, $f(y')$, through the transformed pixels $\mathbf{P'}$.

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 $\mathbf{p'_i} = \mathrm{H}\mathbf{p_i} = [-, y'_i, 1]^T$ and evaluated as: $x_i'^* = f(y'_i)$. Note that the x-value is of no importance and indicated with '-'. By re-projecting this point $\mathbf{p_i'^*} = [x_i'^*, y'_i, 1]^T$ into the original image space we get: $\mathbf{p_i^*} = \mathrm{H}^{-1}\mathbf{p_i'^*}$ with $\mathbf{p_i^*} = [x_i^*, y_i, 1]^T$. Hence, we can evaluate the x-values at different y positions. The algorithm for lane fitting is given in Algorithm 11 utilizing Algorithm 10.

Loss function: The following loss function was constructed to train H-Net. Given M ground-truth lane points $\mathbf{p_i} = [x_i, y_i, 1]^T \in \mathbf{P}$ where $|\mathbf{P}| = \mathrm{M}$, the points are first transformed using the output of H-Net, H. This is shown in Equation (3.2):

$$\mathbf{P'} = \mathrm{H}\mathbf{P} \tag{3.2}$$

with $\mathbf{p_i'} = [x_i', y_i', 1]^T \in \mathbf{P'}$. Through these projected points, we fit a polynomial $f(y') = \mu y'^2 + \nu y' + \rho$ using the least squares closed-form solution as shown in Equation (3.3):

$$\mathbf{w} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{x'} \tag{3.3}$$

with $\mathbf{w} = [\mu, \nu, \rho]^T$, $\mathbf{x'} = [x_1', x_2', ..., x_M']^T$ and

$$\mathbf{Y} = \begin{bmatrix} y_1'^2 & y_1' & 1 \\ \vdots & \vdots & \vdots \\ y_M'^2 & y_M' & 1 \end{bmatrix}$$

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: $\mathbf{p_i^*} = \mathrm{H}^{-1}\mathbf{p_i'^*}$ with $\mathbf{p_i^*} = [x_i^*, y_i, 1]^T$ and $\mathbf{p_i'^*} = [x_i'^*, y_i', 1]^T$. The loss function for projecting points back to the original image space is shown in Equation (3.4).

$$Loss = \frac{1}{M}\sum_{i=1,M}(x_i^* - x_i)^2 \tag{3.4}$$

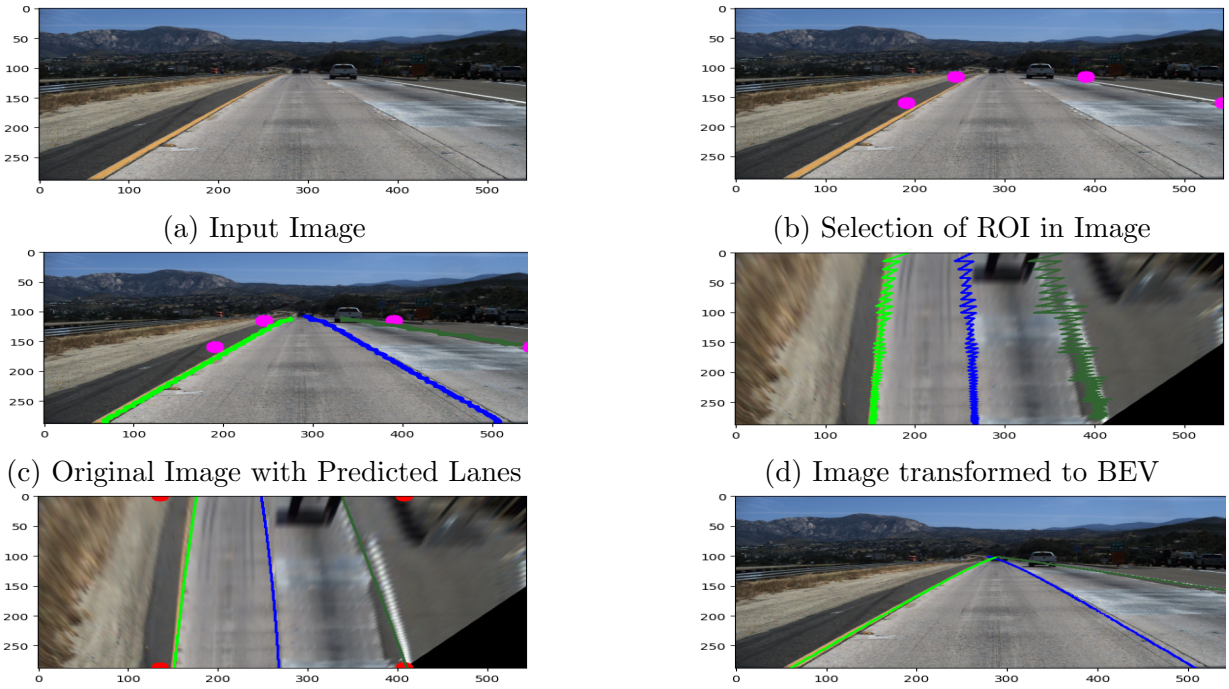
Since the lane fitting is done by using the closed-form solution of the least squares algorithm, the loss is differentiable. We use automatic differentiation to calculate the gradients.

Notice that to convert the image from BEV back to the original image space, we used: $\mathbf{p_i^*} = \mathrm{H}^{-1}\mathbf{p_i'^*}$. When we attempted to implement H-Net, we noticed that during training of H-Net, H-Net outputted matrices that were non-invertible. The consequence was that we couldn't convert from BEV back to the original space. Thus we couldn't calculate the loss and training of the network was halted. Hence we decide to circumvent this issue by skipping the lane fitting step entirely. Despite skipping this step, we still had adequate performance for both LaneNet and ILaneNet.

3.3 Selection of Region of Interest (ROI)

Lane detection methods often employ a two-step pipeline, involving lane candidate generation and subsequent fitting of lane curves. Fitting models such as cubic polynomials, splines, and clothoids are commonly used to describe the lane curves parametrically. This curve fitting process is crucial not only for simplifying the lane curve description but also for enhancing detection accuracy by eliminating outliers.

While CNN-based methods focus on predicting lane candidate positions, the task of fitting these candidates into lane curves is typically accomplished using non-learning methods. Directly fitting lane curves in the image space may not yield optimal results due to perspective distortion. A more effective approach involves transforming the original image into a Bird's Eye View (BEV) through inverse perspective projection and performing curve fitting in this transformed space. This process is shown in Figure 3.2



(a) Input Image

(b) Selection of ROI in Image

(c) Original Image with Predicted Lanes

(d) Image transformed to BEV

(e) Transformed Image with Lane Fitted

(f) Original Image with Fitted Lane

Figure 3.2: Process of Fitting Lanes on an Image

Camera calibration and the calculation of the homographic transformation matrix are typically carried out prior to vehicle movement. However, relying on a fixed homography may fail to adequately address perspective distortion caused by variations in the relative pose between the vehicle and the ground plane (e.g., due to hilly terrain or camera instability), leading to inaccurate lane fitting. Moreover, there are scenarios where pre-calibrating the camera in advance is not feasible.

To resolve this issue, in Lanenet, a neural network called H-Net was utilized and trained using a custom loss function. The purpose of the training was to optimize the network to predict the parameters of a perspective transformation matrix, denoted as H. By accurately

estimating H, the transformed lane points can effectively be fitted with a second or third-order polynomial.

The neural network is designed to take the input image as a condition and generate the projected transformation parameters, enabling the network to adapt to changes in the ground plane. This adaptability ensures that the lane fitting remains accurate even when the ground plane undergoes variations.

The matrix H comprises six degrees of freedom, representing the parameters necessary to perform the perspective transformation. Through training and optimization, the H-Net network learns to predict these parameters, ultimately improving the precision of the lane fitting process.

$$H = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & f & 1 \end{bmatrix}$$

The zeros are placed to enforce the constraint that horizontal lines remain horizontal under the transformation.

In order to train H-Net, the following loss function is constructed. Assuming we have N ground truth points, $\mathbf{p_i} = [x_i, y_i, 1]^T \in P$, a quadratic polynomial of the form $f(y') = \mu y'^2 + \nu y' + \rho$ using the least squares closed-form solution:

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

with $\mathbf{w} = [\mu, \nu, \rho]^T$, $\mathbf{x}' = [x'_1, x'_2, ..., x'_N]^T$ 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. By evaluating the fitted polynomial at each location y'_i , a prediction x'^*_i is obtained. Subsequently, these predictions are projected back: using the $\mathbf{p_i^*} = \mathrm{H}^{-1}\mathbf{p_i'^*}$ 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 and hence automatic differentiation to calculate the gradients.

3.3.1 H-Net Architecture

As discussed earlier, to address the issue of fitting a polynomial through lane pixels in the original image space, a more effective approach is often employed. This involves projecting the image into a bird's-eye view representation, where the lanes appear parallel to each other. In this transformed perspective, curved lanes can be accurately fitted using lower-order polynomials, typically ranging from 2nd to 3rd order. This technique offers a solution to the challenge of accommodating curved lanes without the need for excessively high-order polynomials in the original image space..

To tackle this problem, a neural network called H-Net was trained with a custom loss function. H-Net is optimized end-to-end and tasked with predicting the parameters of a perspective transformation matrix H. This tailored matrix ensures that the transformed lane points can be optimally fitted with 2nd or 3rd order polynomials. Crucially, the network's prediction is conditioned on the input image, enabling it to adapt the projection parameters in the presence of ground-plane changes. This adaptability guarantees accurate lane fitting even when the ground plane undergoes modifications.

H-Net is trained for a 3rd-order polynomial fit, with a scaled version of input image with dimension 128×64. The network is trained using Adam with a batch size of 10 and learning rate 5e-5 until convergence. Table 3.1 presents the network architecture of H-Net.

It's crucial to emphasize that H-Net's network design intentionally maintains a compact size. To reduce input dimensions, it utilizes max pooling. The remaining components of the network consist of successive sequences of 3×3 convolutions, batch normalization, and Rectified Linear Units (ReLUs), ultimately concluding with two fully-connected layers.

3.4 Training of Lanenet

Lanenet is trained using a specific configuration: it employs an embedding dimension of 4, with δ_v set to 0.5 and δ_d set to 3. The input images are rescaled to a fixed size of 512×256. During training, the network utilizes the Adam optimizer with a batch size of 8 and a learning rate of 5e-4. The training process continues until convergence is reached, ensuring optimal performance. The algorithm and flowchart for training is given in Algorithm 1 and Figure 3.3 respectively.

Type	Filters	Size/Stride	Output
Conv+BN+ReLU	16	3x3	128x64
Conv+BN+ReLU	16	3x3	128x64
Maxpool		2x2/2	64x32
Conv+BN+ReLU	32	3x3	64x32
Conv+BN+ReLU	32	3x3	64x32
Maxpool		2x2/2	32x16
Conv+BN+ReLU	64	3x3	32x16
Conv+BN+ReLU	64	3x3	32x16
Maxpool		2x2/2	16x8
Linear+BN+ReLU		1x1	1024
Linear		1x1	6

Table 3.1: H-Net network architecture.

Algorithm 1 Lanenet Model

Input:

U : Input Images
Y : Binary Segmentation Image
Z : Instance Segmentation Image

procedure
    $R \leftarrow$ Resize(U)
    for $i \leftarrow 1$ to *epochs* **do**
        $A \leftarrow$ Shared Encoder(R)
        $B \leftarrow$ Embedding Branch(A)
        $C \leftarrow$ Segmentation Branch(A)
        $L1 \leftarrow$ Compute Embedding Loss(B, Z)
        $L2 \leftarrow$ Compute Segmentation Loss(C, Y)
        Update Weights to Minimize Losses
    end for
end procedure

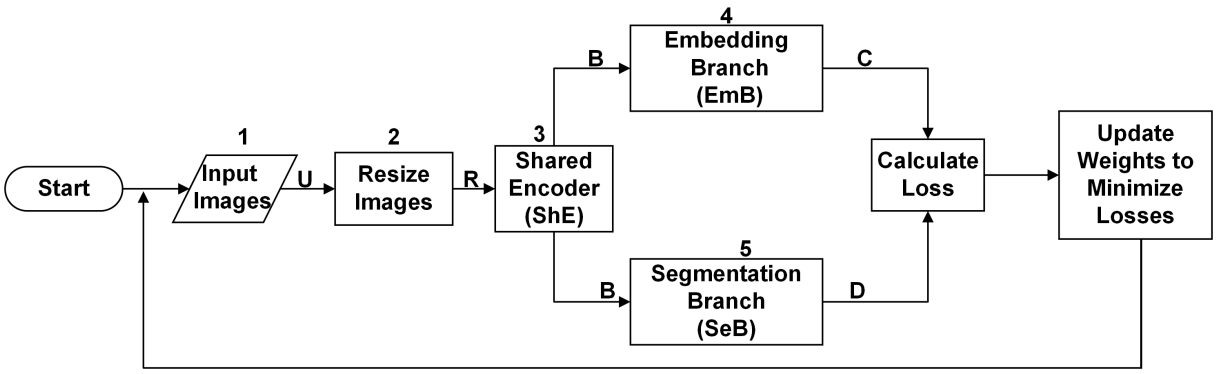


Figure 3.3: Flowchart for training LaneNet

3.5 Testing of Lanenet

In the lane detection process using LaneNet, after the image is provided as an input to the trained embedding and segmentation model, two outputs are obtained: a binary segmentation map and an embedding map.

The binary segmentation map indicates which pixels in the image belong to the lane and which pixels do not. It assigns a binary value (typically 1 or 0) to each pixel, effectively separating the lane region from the background.

Simultaneously, the embedding map is generated, which assigns a vector or feature representation to each pixel in the image. This embedding encodes information about the pixel's position and characteristics relevant to lane detection.

Next, the lane pixels are clustered into different instances based on their embeddings. This clustering step groups pixels that belong to the same lane, allowing the algorithm to distinguish multiple lanes if present in the image.

Once the lane pixels are separated into distinct instances, a curve fitting process is applied to each instance. This involves fitting a curve or line to the clustered pixels, aiming to accurately represent the lane trajectory.

Finally, a spline representation is outputted for each lane, which provides a smooth and continuous estimation of the lane's shape. Splines are commonly used to describe curves in a flexible and precise manner, allowing for smooth lane boundaries to be generated. This is shown in Figure 3.4.

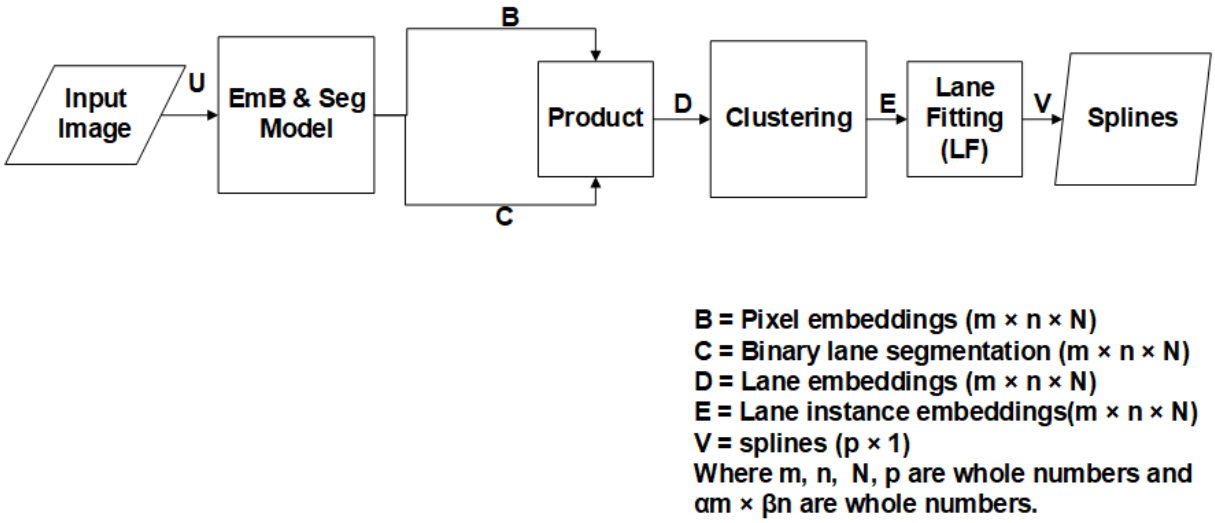


Figure 3.4: Pipeline for Testing an Image in Lanenet

3.5.1 Improved Lanenet

To add the number of lanes (NoL) to the input image, we were inspired by the Conditional GAN network[6]. Hence, we first one-hot encoded the NoL. The one-hot encoded representation of the number of lanes was subsequently fed into a fully connected (FC) layer. This FC layer processes the lane information and produces an output with dimensions that can be later reshaped for concatenation with the input image. An FC layer connects every input neuron to every output neuron. Finally, the FC layer output was reshaped and concatenated with the original image. This now becomes the input to the LaneNet model and this new modification to the network is called ILaneNet.

To illustrate this process, assume that we have an input image of size $1280 \times 720 \times 3$ (αm width $\times$ βn height $\times c$ channels) and aim to rescale it to a target size of $512 \times 256 \times 3$ (m width $\times$ n height $\times$ c channels). In the LaneNet approach, we directly scale the image to the target size of $512 \times 256 \times 3$ (m width $\times$ n height $\times$ c channels).

However, in the ILaneNet approach, the image is scaled to a slightly different shape, specifically, $512 \times 255 \times 3$ (m width $\times$ $(n-1)$ height $\times$ c channels). The lane information is one-hot encoded and then passed through a fully connected layer with a size of $512 \times 3 = 1536$ ($m \cdot c$). We assumed that the possible lane values range from 1 to $m \cdot c$ (where $m \cdot c$ is the maximum possible value). Hence, our one-hot encoding was performed as follows: 1 lane can be encoded as $[1, 0, 0, \ldots, 0]$, 2 lanes can be encoded as $[0, 1, 0, \ldots, 0]$, 3 lanes

can be encoded as $[0, 0, 1, \ldots, 0]$ and $m \cdot c$ lanes can be encoded as $[0, 0, 0, \ldots, 1]$. In this encoding, each position in the array corresponds to a possible number of lanes, and only one position is "hot" (set to 1) while the others are "cold" (set to 0) to indicate the specific value of the number of lanes.

After passing the encoded lanes through a fully connected layer, it is reshaped into a tensor of size $512 \times 1 \times 3$ (m width $\times 1$ height $\times c$ channels). The reshaped lane information is then concatenated with the rescaled image, which has a size of $512 \times 255 \times 3$, to form an output of size $512 \times 256 \times 3$. This results in the same shape as the rescaled image used in the LaneNet approach. The output of concatenation would now be used as the input tensor for the LaneNet network.

The ILaneNet architecture is depicted in Figure 3.6. The intricate process of merging the inputs is detailed in Figure 3.5. We have also provided a pseudocode detailing how the lanes and input image are merged in Algorithm 2.

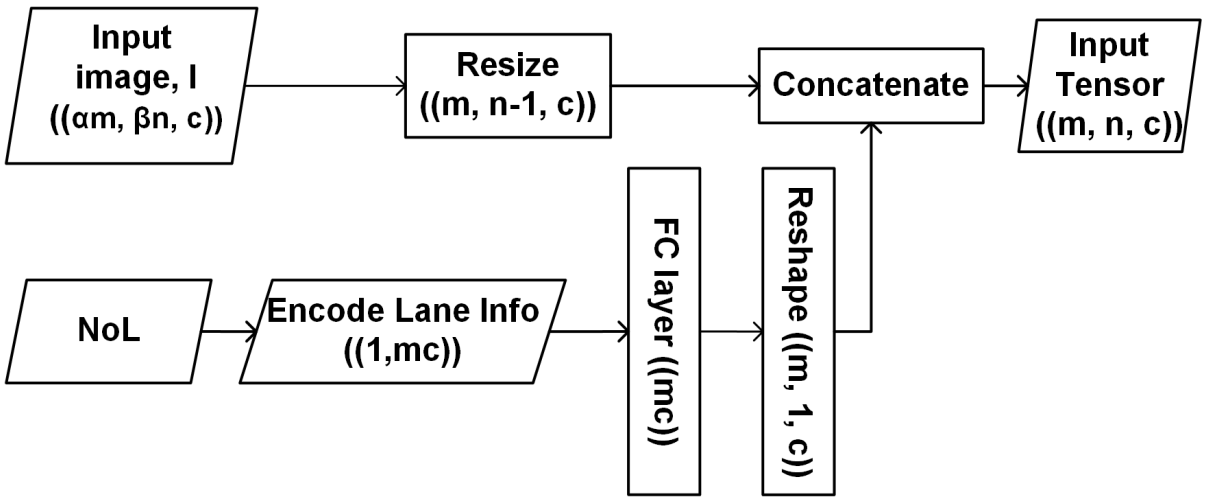


Figure 3.5: Concatenating image with lanes

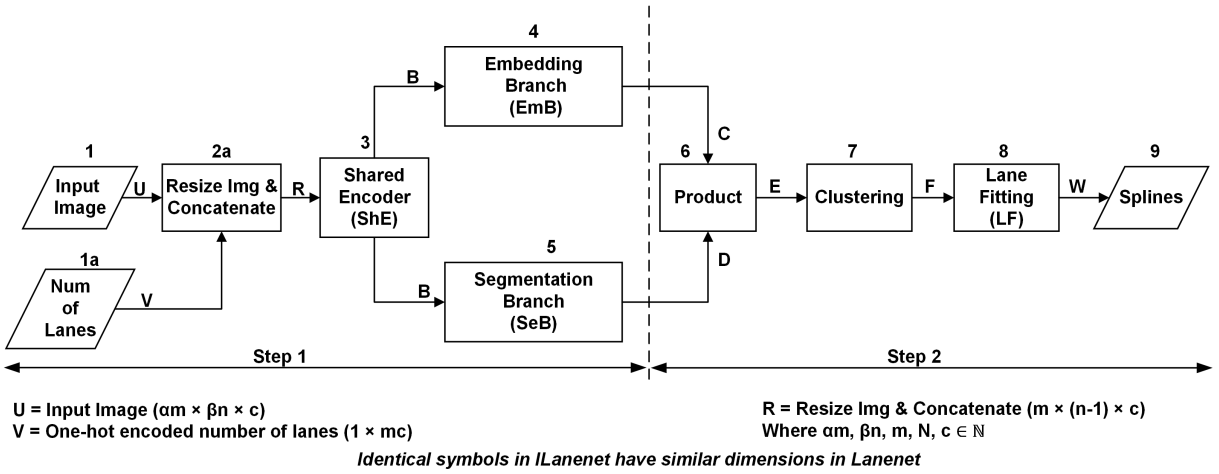


Figure 3.6: ILanenet Architecture

Algorithm 2 Add Lane Information

Input:

I : Input image (αm × βn × c)
NoL : Number of Lanes where NoL > 0
m : Width of the target image
c : Number of channels in input image
n : Height of the target image

Output:

outImg : Image with added lane information (m × n × c)

1: **procedure**
2: // Resize the input image to m × (n − 1) × c
3: $resizedImg \leftarrow$ ResizeImage(I, (m, n-1, c))
4: // One-hot encode lane information
5: $laneInfo \leftarrow$ EncodeLaneInformation(NoL)
6: // Extract lane information using FCN network
7: $extractedInfo \leftarrow$ FCN(laneInfo, size=mc)
8: // Reshape output

Algorithm 2 Add Lane Information Continued

```
    laneInfo ← Reshape(extractedInfo, (m, 1, c))
        // Concatenate lane information with the image
    outImg ← Concatenate(resizedImg, laneInfo)
    return outImg
end procedure
```

3.6 New Performance Metrics

We introduce a new performance metric, which we refer to as the capacity of the lane detection system. The expression for capacity is given in Equation 3.5. The term "capacity of the lane detection system" offers a novel perspective on understanding the accuracy of a lane detection system.

To illustrate this idea, consider a scenario with two lanes on the road. If the lane detection system can accurately identify both of these lanes, it signifies that more vehicles can smoothly traverse the road. In essence, this means the lane detection system possesses a higher capacity because it enables the full utilization of available lanes, thereby promoting efficient traffic flow.

Conversely, if the system can only detect one lane accurately while overlooking the other, it implies that all vehicles may be restricted to a single portion of the road. This can lead to suboptimal road usage, potentially resulting in traffic congestion. In such cases, the lane detection system is deemed to have a lower capacity because it cannot effectively harness all the lanes, which in turn limits traffic efficiency.

In this context, the assessment of capacity takes a global perspective, considering a network of autonomous vehicles whose paths are centrally planned. The emphasis is on the system's ability to orchestrate the movement of multiple vehicles, ensuring they navigate the road with optimal efficiency by leveraging the full spectrum of available lanes.

We therefore define the term "capacity" as the system's ability to detect and effectively utilize existing lane markings/lanes on the road. Conversely, lost capacity refers to the system's inability to effectively utilize existing lane markings/ lanes on the road.

In addition to capacity and lost capacity, we also introduce a new metric known as unsafe driving measure. The unsafe driving measure evaluates the extent to which the lane detection system may provide inaccurate lane predictions that could lead the driver to make unsafe decisions. When the system generates false positives, it mistakenly indicates the presence of a lane where there isn't one. These false positives can mislead the driver

into believing that a section of the road is a legitimate lane. As a result, an autonomous vehicle may attempt to maneuver into this supposed lane, thinking it's safe to do so. This leads to unsafe driving. The expressions for capacity, lost capacity and unsafe driving measure are given in Equations 3.5, 3.6 and 3.7 respectively:

$$\text{Capacity}^l = \frac{TP^l}{TP^l + FN^l} \tag{3.5}$$

$$\text{Lost Capacity}^l = 1 - \text{Capacity}^l \tag{3.6}$$

$$\text{Unsafe Driving Measure}^l = \frac{FP^l}{TP^l + FN^l} \tag{3.7}$$

with TP^l being the number of correctly predicted lanes, FP^l being the number of wrongly predicted lanes and FN^l being the number of missed ground-truth lanes. It is also worth noting that the expression for capacity and unsafe driving measure coincide with the expression for recall and false positive score.

When abstracting a lane as a line, it may not provide sufficient information for practical use. Imagine a scenario where the network outputs only one lane line. In this case, the system lacks crucial details about the lane's boundaries. This makes it challenging for a vehicle to determine where it should pass. To address this limitation, we introduce a more advanced lane abstraction approach.

In the lane abstraction approach, instead of focusing solely on individual lane lines, we consider entire lanes as distinct entities. This means that we count the number of lanes present in the scene, providing a more comprehensive representation of the road layout. This approach is valuable because it offers a clearer understanding of the road configuration, enabling vehicles to make more informed decisions regarding lane changes and safe navigation.

To better suit the lane abstraction approach, modifications were introduced to the following equations to consider lanes rather than lines. The expressions for false positive score and false negative score in the lane abstraction approach are given in Equations (3.8) and (3.9) respectively.

$$FPS^L = \frac{F^L_{pred}}{N^L_{pred}} \tag{3.8}$$

$$FNS^L = \frac{M^L_{pred}}{N^L_{gt}} \tag{3.9}$$

with F^L_{pred} being the number of wrongly predicted lanes, N^L_{pred} the number of predicted lanes, M^L_{pred} the number of missed ground-truth lanes and N^L_{gt} being the number of all ground-truth lanes. The expressions for capacity, lost capacity and unsafe driving measure in the lane abstraction approach are given in Equations 3.10, 3.11 and 3.12 respectively:

$$\text{Capacity}^L = \frac{TP^L}{TP^L + FN^L} \tag{3.10}$$

$$\text{Lost Capacity}^L = 1 - \text{Capacity}^L \tag{3.11}$$

$$\text{Unsafe Driving Measure} = \frac{FP^L}{TP^L + FN^L} \tag{3.12}$$

with TP^L being the number of correctly predicted lanes, FP^L being the number of wrongly predicted lanes and FN^L being the number of missed ground-truth lanes.

Chapter 4

Experimentation and Results

The experiments conducted in this thesis were thoughtfully designed to align perfectly with the stated objectives. The primary focus was to evaluate the performance of two lane detection methods, namely LaneNet and ILaneNet, while also visualizing and analyzing the obtained lane detection results. Furthermore, the investigation of the limitations associated with the proposed ILaneNet was another critical aspect of the research. The remaining sections describe the datasets used and the experiments that were performed.

4.1 Dataset

The TuSimple dataset was chosen as our primary dataset for several compelling reasons. To begin with, the dataset consists of 6,408 road images captured on US highways, with each image having a resolution of 1280×720. This resolution provides a substantial amount of detail and context, which is crucial for training a robust computer vision model.

Next, the dataset's size is substantial, with 3,626 images allocated for training, 358 for validation, and 2,782 for testing. This ample number of images is vital for effectively training our neural network. A larger dataset can help prevent overfitting, as the model has access to a diverse range of road scenarios, ensuring it can generalize well to unseen data.

Additionally, we employed data augmentation techniques to further expand the training dataset. Data augmentation involves applying various transformations such as rotation, flipping, and scaling to the existing images, effectively creating new samples. This process

enhances the model's ability to handle different road conditions and perspectives, making it more robust to variations encountered during real-world deployment.

One key aspect of our dataset selection was having diversity in road conditions. This diversity was found in the TuSimple dataset which contains road images under various weather conditions. This diversity is essential for assessing the model's generalization capabilities and performance in adverse scenarios. By evaluating the model on real-life data with different weather conditions, we can gain a better understanding of its reliability and readiness for deployment in practical environments.

Moreover, the TuSimple dataset being a real-life dataset is a significant advantage. Real-life datasets are more likely to include challenging scenarios and situations that might not be fully represented in synthetic or simulated datasets. This aspect makes the TuSimple dataset more suitable for training and evaluating a real-world application, as it ensures our model is exposed to a wide range of complexities and challenges found in actual highway scenes.

The factors listed above make the TuSimple dataset a fitting choice for our project.

4.2 Hyperparameter Tuning

4.2.1 Selection of Embedding Dimensionality for LaneNet and ILaneNet

In our pursuit to enhance the performance of lane detection networks, we recognized the significance of selecting an appropriate number of dimensions for the embedding layer in both LaneNet and ILaneNet. As explained earlier, the embedding layer plays a critical role in representing information about the specific lane instance. The choice of the embedding dimensionality is a delicate balance between achieving higher model expressiveness while avoiding overfitting and computational inefficiencies. Higher dimensions allow for increased feature richness and complexity, potentially leading to improved lane detection results. However, excessively high-dimensional embeddings can lead to overfitting, where the model becomes overly specialized to the training data and performs poorly on unseen data. Similarly, lower-dimensional embeddings may be computationally efficient but might lose important discriminative information, resulting in reduced accuracy. Our objective was to find the optimal number of dimensions that strikes a balance between computational efficiency, expressiveness and generalization.

To address this crucial aspect of our research, we conducted experiments to evaluate different embedding dimensionality settings. We systematically varied the number of dimensions and assessed the model's performance across a range of metrics, including accuracy, false negative score and false positive score. The results for the experiments are summarized in Table 4.1.

Table 4.1: Comparison of embedding dimensionality for LaneNet

Embedding Dimension	**Accuracy**	**FPS[I]**	**FNS[I]**
1	84.96	44.1	37.17
2	92.81	23.5	11.3
4	92.3	23	11.1
5	92.9	22.9	10.9
7	93.2	21.9	10.0
9	92.6	22.6	11.8

From Table 4.1, we can see that an embedding dimension of 1 produces the least accuracy of 84.96% with the worst false positive score and the worst false negative score of 44.1% and 37.17% respectively. This shows that an embedding dimension of 1 is inadequate for capturing information about the lane instance a pixel belongs to. We also observe from the table that as the embedding dimension increases from 2 to 7, we get a corresponding increase in better results for accuracy, false positive score and false negative score. This can be attributed to the embedding dimension proving sufficient parameters to better capture information about the specific lane instance a lane belongs to. From an embedding dimension of 9, we start to see that the accuracy, false positive and false negative score starts to drop. This could be the result of the model having more than enough parameters and tending to become more specialized to the training data.

Considering that there is at most 1% difference between accuracy, false positive score and false negative score for an embedding dimension from 2 to 7, we chose an embedding dimension of 4 because it was the most computationally efficient.

4.2.2 Impact of Minimum Area Threshold on Lane Detection Performance

In the experimentation section of our research, we investigated the influence of the minimum area threshold on the performance of our lane detection algorithm. This parameter

plays a critical role in filtering out small and irrelevant regions from the binary segmentation output, with the aim of improving the accuracy and robustness of the lane detection process.

The minimum area threshold is a user-defined parameter that specifies the minimum allowable area for a component to be considered a valid lane region. After applying image morphology operations and connected component analysis to the binary segmentation result, each identified component's area is evaluated.

We conducted a series of experiments where we varied the minimum area threshold over a range of values.

Through these experiments, we aimed to find the optimal minimum area threshold that strikes a balance between removing false positives and preserving genuine lane markings. A threshold that is too high might filter out small but valid lane regions, leading to missed detections and reduced recall. On the other hand, a threshold that is too low may allow noise and artifacts to be considered as valid lanes, reducing the overall accuracy and precision of the lane detection system.

By systematically evaluating the minimum area threshold, we were able to identify that a threshold hold value of 100 yielded the best lane detection performance. This optimal value provided an effective trade-off between noise removal and lane preservation, leading to accurate and reliable lane detection results.

Furthermore, we investigated the impact of the minimum area threshold on the algorithm's computational efficiency. Setting a higher threshold generally reduces the number of components to process, potentially speeding up the lane detection process. However, an excessively high threshold might remove genuine lane markings, necessitating a careful balance between accuracy and computational cost.

The results and analyses obtained from these experiments contributed valuable insights into the significance of the minimum area threshold in lane detection. The findings guided us in determining the best threshold setting to achieve superior lane detection performance, which is crucial for real-world applications such as autonomous driving and driver-assistance systems.

4.2.3 Selection of Backbone and Clustering Algorithm

In this section of the experimentation, we focused on the crucial task of selecting the most suitable backbone and clustering algorithm for LaneNet. A backbone network is used for feature extraction. Clustering plays a fundamental role in grouping lane feature points and

identifying lane boundaries accurately. The performance and effectiveness of the clustering algorithm significantly impact the overall quality of lane detection results. The results of our experiments are given in Table 4.2.

Table 4.2: Comparison of Backbone and Clustering Methods for LaneNet

Backbone	**Clustering**	**Accuracy**
BiseNetV2	DBScan	52.23%
ENet	Meanshift	92.3%

To make an informed decision, we evaluated multiple clustering algorithms, each with its unique strengths and characteristics. The selected algorithms for comparison included K-means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Mean Shift clustering.

Each clustering algorithm was applied to the lane embeddings extracted from the images, and the ability for the clustering algorithm to distinguish between lanes was assessed and compared. The accuracy of lane detection was computed to objectively evaluate the effectiveness of each clustering method. Additionally, visual inspection of the lane detection results was performed to gain qualitative insights into the clustering performance.

KMeans was the least suitable algorithm for lane embeddings based on visual evidence in ILaneNet. In LaneNet, since we don't supply the NoL, we can't use the KMeans algorithm since we don't know what the number of clusters will be. In ILaneNet, we possess the knowledge of the number of lanes, enabling us to specify the number of clusters to be provided to the KMeans algorithm.he inherent problem is rooted in the fundamental operation of KMeans, which involves the random selection of an embedding to establish clusters. The inherent problem with this approach lies in the fact that KMeans often selects multiple embeddings from the same lane, leading to the erroneous grouping of a single lane into multiple clusters.

DBScan was found to perform better compared to KMeans when the same backbone network was used but worse compared to the Meanshift algorithm with ENet backbone. One advantage of DBScan is that there is no requirement to specify the number of clusters and hence the approach can be used for both LaneNet and ILaneNet. However, selecting parameters for DBScan namely the epsilon value and the mininum number of points can be challenging. An epsilon value of 0.35 with minimum number of samples of 1000 was found to work well for DBScan.

The Mean Shift clustering algorithm was found to work best with the ENet backbone.

Mean shift clustering algorithm, a non-parametric approach, allows for adaptive density estimation and is well-suited for datasets with varying densities.

The results of our experiments shed light on the strengths and weaknesses of each clustering algorithm for lane detection. We identified the algorithm that yielded the most accurate and reliable lane boundaries across diverse driving scenarios. Additionally, we analyzed the computational performance of each algorithm, ensuring the selected method strikes a balance between accuracy and efficiency.

4.2.4 Fixed Homography vs Conditional Homography

We also conducted experiments where lanes were fitted with two different approaches: fixed homography and conditional homography, both of which were implemented using the DBScan clustering algorithm. In fixed homography, we chose a fixed perspective transformation matrix for all images in the test set. In conditional homography, while we did not incorporate H-Net into our experiments, we successfully derived the transformation matrix for each image within the LaneNet framework, as described in [37], enabling us to apply perspective transformation within LaneNet. The results of these experiments are presented in Table 4.3.

Table 4.3: LaneNet Fixed Homography vs Conditional Homography

Backbone	**Clustering Algorithm**	**Homography Type**	**Accuracy (%)**
BiseNetV2	DBScan	Fixed Homography	52.23
BiseNetV2	DBScan	Conditional Homography	65.49

Referring to Table 4.3, it can be observed that using conditional homography led to a notable accuracy improvement of 13.26%. This outcome aligns with the findings from the initial LaneNet implementation, where H-Net was employed to generate the perspective transform matrix, and conditional homography similarly yielded accuracy gains. The rationale behind this accuracy enhancement lies in the fact that conditional homography takes into consideration groundplane variations and leverages this data to make an informed selection of an appropriate perspective matrix.

4.3 Results

The final configuration for LaneNet and ILaneNet used E-net as a backbone together with the Meanshift algorithm. The results of running LaneNet and ILaneNet on the TuSimple

Dataset are given in Tables 4.4, 4.5 and 4.6. Additionally, the reasons for the results are also explained in this section.

Table 4.4: Lane Abstraction

NETWORK	USED CAP.	LOST CAP.	UNSAFE DRIV. MEASURE
ILaneNet	87.5 %	12.5 %	27.3 %
LaneNet	80.4 %	19.6 %	38.5 %

Table 4.5: Line Abstraction

NETWORK	USED CAP.	LOST CAP.	UNSAFE DRIV. MEASURE	ACC
ILaneNet	93.1 %	6.9 %	13.9 %	94.5 %
LaneNet	88.9 %	11.1 %	23.0 %	92.3 %

Table 4.6: Speed metrics

Metric	LaneNet	ILaneNet
Forward pass time per image (ms)	43.5	51.6
Clustering time per image (ms)	231.8	232.8
Total time per image (ms)	275.34	284.4

Quantitative Analysis: The outcome of running the experiments are given in Table 4.4 and Table 4.5.

Line Abstraction For line abstraction, it is evident that ILaneNet is superior to LaneNet across all metrics. ILaneNet exhibits slightly higher accuracy compared to LaneNet, implying that ILaneNet correctly classifies a slightly greater number of data points than LaneNet. However, this improvement is so slight that it may not be considered a significant advantage of ILaneNet over LaneNet.

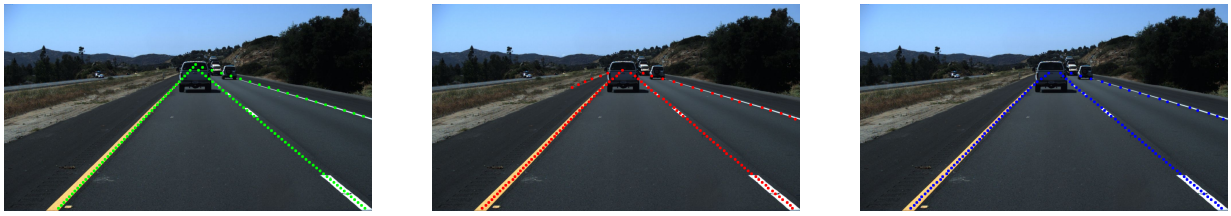
ILaneNet's greatest strengths over LaneNet lie in terms of reducing false positives and false negatives, resulting in a higher capacity (recall) for the lane detection system. ILaneNet achieves a recall of approximately 93.1%, meaning it's more likely to identify all existing lane lines compared to LaneNet. In terms of capacity, this suggests that ILaneNet is more likely to have vehicles utilizing all available lanes, reducing the likelihood of lost capacity. This is also reflected in the lost capacity score.

ILaneNet demonstrates an unsafe driving measure (false positive score) of approximately 13.9%, significantly lower than LaneNet's score of 23.0%. This implies that drivers

(a) Ground Truth

(b) LaneNet detects False Positive

(c) ILaneNet does not detect False Positive



(d) Ground Truth

(e) LaneNet detects False Positive

(f) ILaneNet does not detect False Positive

(g) Ground Truth

(h) LaneNet misses a Lane

(i) ILaneNet identifies the missed Lane

Figure 4.1: Lane Detection in LaneNet and ILaneNet

utilizing LaneNet are more likely to identify false lanes as real lanes, potentially leading to unsafe maneuvers compared to drivers utilizing ILaneNet.

Lane Abstraction As we discussed earlier, assessing metrics such as used capacity, lost capacity, and unsafe driving measures in terms of lanes provides more meaningful insights compared to assessing them in terms of individual lines. The results for lane-based assessments are presented in Table 4.4. In this table, it is evident that ILaneNet outperforms LaneNet across these crucial metrics. Specifically: ILaneNet achieves a higher used capacity of 87.5% compared to LaneNet's 80.4%. This signifies that ILaneNet is more proficient at identifying lanes compared to LaneNet. Consequently, ILaneNet also exhibits a lower

lost capacity compared to LaneNet.

ILaneNet demonstrates a lower unsafe driving measure, with a score of 27.3%, in contrast to LaneNet's score of 38.5%. This implies that vehicles utilizing ILaneNet are less likely to result in unsafe driving compared to LaneNet.

Visual Analysis: From Figure 4.1 presented above, we can gain some insight into the lane detection capabilities of LaneNet and ILaneNet.

In the context of false positive detection, the ground truth images (Figures 4.1a and 4.1d) serve as the baseline, representing the actual lane markings. LaneNet's performance, as shown in Figures 4.1b and 4.1e, reveal that it tends to detect additional, false positive lane markings not present in the ground truth. This suggests that LaneNet may have a tendency to over-detect lanes in certain scenarios. Conversely, ILaneNet's results in Figures 4.1c and 4.1f demonstrate that it is more conservative in its lane detection approach. ILaneNet does not detect these false positive lane markings, which is advantageous when accuracy and avoiding false alarms are paramount.

Additionally, when considering missed lane detection, Fig. 4.1g represents the ground truth with all the lane markings correctly annotated. However, Figure 4.1h shows that LaneNet misses one of the lane markings present in the ground truth. This indicates that LaneNet may have limitations in accurately identifying all lane markings. In contrast, Figure 4.1 i illustrates ILaneNet's ability to successfully identify the missed lane marking, showcasing its strength in capturing lane markings that may be overlooked by other algorithms.

These sample observations give an insight into the strengths of each algorithm. LaneNet overall has a higher false positive rate and also tends to miss lane markings. Conversely, ILaneNet excels in capturing missed lane markings while avoiding false positives. This makes ILaneNet a better model compared to LaneNet overall.

Speed Metrics: The speed metrics for both LaneNet and ILaneNet are given in Table 4.6. From Table 4.6, we notice that LaneNet takes a shorter time of 275.34ms compared to ILaneNet which takes a slightly longer time of 284.4ms to process an image. This makes sense because additional time will be required to merge the total number of lanes with the input image in ILaneNet. Once the total number of lanes have been merged with the input image, we can see that it takes a similar amount of time to cluster both images. In conclusion, having an additional time of about 10ms to process the image in ILaneNet is a small price to pay for the significant advantage that ILaneNet gives in reducing false negatives and false positives.

Chapter 5

Conclusion and Future Work

In conclusion, this thesis introduced an enhanced lane detection model, ILanenet, designed to provide robust and accurate lane detection in a variety of driving scenarios. By incorporating multiple inputs, including the driving scene and the number of lanes, and leveraging a fully connected layer for feature extraction, ILanenet demonstrated improved performance compared to its predecessor, Lanenet. It showcased the ability to avoid false lane recognition and maintain accuracy across diverse road conditions.

In addition, we introduced new performance evaluation metrics namely capacity, lost capacity and unsafe driving measure which offer a more comprehensive and nuanced assessment of lane detection techniques. Furthermore, we also introduced a shift in the evaluation paradigm by advocating the use of a lane abstraction approach for assessing detected lanes, diverging from the conventional line abstraction methodology. We also compared the time to process images in both Lanenet and ILanenet and found that ILanenet takes slightly more time to process an image compared to Lanenet. However, this is a good tradeoff to make if we look at the benefit ILanenet provides over Lanenet.

While this thesis marks a significant step forward in the field of lane detection, there are several promising avenues for future work and research. Firstly, we plan to further enhance ILanenet by utilizing information from the number of lanes to extrapolate the positions of missing lanes, addressing scenarios where lane markings may be obscured or incomplete. This improvement will contribute to a more comprehensive and precise lane detection system.

Additionally, the elimination of false positives remains a critical focus of our future work. We intend to refine ILanenet's architecture and training methods to minimize the

occurrence of false lane detections, thereby enhancing its reliability for real-world applications. This will be vital for ensuring the safety and effectiveness of autonomous vehicles and advanced driver-assistance systems.

Furthermore, we aim to explore the integration of ILanenet with other advanced technologies, such as simultaneous localization and mapping (SLAM) and object detection, to provide a more holistic understanding of the driving environment. This multidimensional approach will contribute to a more comprehensive and context-aware perception system for autonomous vehicles.

In summary, this thesis has laid the foundation for a more robust and accurate lane detection system, ILanenet, and has outlined a roadmap for future research, focusing on extrapolating missing lanes, eliminating false positives, and integrating advanced technologies to enhance the capabilities of ILanenet in the dynamic field of autonomous driving and computer vision. These efforts will contribute to safer and more reliable autonomous driving systems and further advance the state of the art in lane detection technology.

References

[1] TuSimple Benchmark. https://github.com/TuSimple/tusimple-benchmark. Accessed: Insert Date Here.

[2] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[3] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[4] Plamen P. Angelov. *Empirical Approach to Machine Learning*. Studies in Computational Intelligence, 800. Springer International Publishing, Cham, 1st ed. 2019. edition, 2019.

[5] Nicholas Apostoloff and Alexander Zelinsky. Vision in and out of vehicles: Integrated driver and road scene monitoring. *I. J. Robotic Res.*, 23:513–538, 04 2004.

[6] Rowel Atienza. *Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more*. Packt Publishing Ltd, 2020.

[7] Automotive - Technical articles - TI E2E support forums. What's next for advanced driver assistance systems?, 2019. Accessed: 11-Nov-2019.

[8] Zhibin Bao, Sabir Hossain, Haoxiang Lang, and Xianke Lin. A review of high-definition map creation methods for autonomous driving. *Engineering Applications of Artificial Intelligence*, 122:106125, 2023.

[9] Monowar H Bhuyan, D. K Bhattacharyya, and J. K Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications surveys and tutorials*, 16(1):303–336, 2014.

[10] Srivalli Boddupalli, Akash Someshwar Rao, and Sandip Ray. Resilient cooperative adaptive cruise control for autonomous vehicles using machine learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[11] Jasmin Breitenstein, Jan-Aike Termöhlen, Daniel Lipinski, and Tim Fingscheidt. Systematization of corner cases for visual perception in automated driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1257–1264, 2020.

[12] Long Chen, Qingquan Li, Qingzhou Mao, and Qin Zou. Block-constraint line scanning method for lane detection. In *IEEE Intelligent Vehicles Symposium*, pages 89–94, 2010.

[13] Nandan Bangalore Chetan, Jiayuan Gong, Haiying Zhou, Dong Bi, Jianping Lan, and Leipeng Qie. An overview of recent progress of lane detection for autonomous driving. In *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*, pages 341–346, 2020.

[14] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158, 2016.

[15] B De Brabandere, D Neven, and L Van Gool. Corr abs/1708.02551, 2017.[6] l. *Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. CoRR abs/1606.00915*, 2016.

[16] Daniel J. Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[17] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70:447–489, 2019.

[18] Krist V. Gernaey, Jakob K. Huusom, and R. Gani. *12th International Symposium on Process Systems Engineering and 25th European symposium on Computer Aided Process Engineering*. Elsevier, 2015.

[19] Mohsen Ghafoorian, Cedric Nugteren, Nora Baka, Olaf Booij, and Michael Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[20] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[22] AB. Hillel, R. Lerner, D. Levi, and G. Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, 2014.

[23] Qiao Huang and Jinlong Liu. Practical limitations of lane detection algorithm based on hough transform in challenging scenarios. *International Journal of Advanced Robotic Systems*, 18(2):17298814211008752, 2021.

[24] Abdul Rehman Javed, Muhammad Usman, Saif Ur Rehman, Mohib Ullah Khan, and Mohammad Sayad Haghighi. Anomaly detection in automated vehicles using multi-stage attention-based convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4291–4300, 2021.

[25] Donald Knuth. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[26] Hui Kong, Jean-Yves Audibert, and Jean Ponce. Vanishing point detection for road detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 96–103, 2009.

[27] Yassin Kortli, Souhir Gabsi, Lew F.C. Lew Yan Voon, Maher Jridi, Mehrez Merzougui, and Mohamed Atri. Deep embedded hybrid cnn–lstm network for lane detection on nvidia jetson xavier nx. *Knowledge-Based Systems*, 240:107941, 2022.

[28] Leslie Lamport. *LaTeX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[29] Chanho Lee and Ji-Hyun Moon. Robust lane detection and tracking for real-time applications. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):4043–4048, 2018.

[30] Seokju Lee, Jun-Sik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In-So Kweon. VPGNet: Vanishing point guided network for lane and road marking detection and recognition. In *International Conference on Computer Vision*, pages 1965–1973, 2017.

[31] Qingquan Li, Long Chen, Ming Li, Shih-Lung Shaw, and Andreas Nüchter. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Transactions on Vehicular Technology*, 63:540–555, 2014.

[32] Qingquan Li, Jian Zhou, Bijun Li, Yuan Guo, and Jinsheng Xiao. Robust lane-detection method for low-speed environments. *Sensors*, 18(12), 2018.

[33] Wenhui Li, Feng Qu, Jialun Liu, Fengdong Sun, and Ying Wang. A lane detection network based on ibn and attention. *Multimedia Tools and Applications*, 79:16473–16486, 2020.

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[35] Yan Liu, Jingwen Wang, Yujie Li, Canlin Li, and Weizheng Zhang. Lane-gan: A robust lane detection network for driver assistance system in high speed and complex road conditions. *Micromachines*, 13(5), 2022.

[36] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. A real-time lane marking localization, tracking and communication system. *Computer Communications*, 73:132–143, 2016.

[37] MaybeShewill-CV. lanenet-lane-detection, 2023. GitHub repository.

[38] Joel C McCall and Mohan M Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE transactions on intelligent transportation systems*, 7(1):20–37, 2006.

[39] Sandipann P. Narote, Pradnya N. Bhujbal, Abbhilasha S. Narote, and Dhiraj M. Dhane. A review of recent advances in lane detection and departure warning system. *Pattern Recognition*, 73:216–234, 2018.

[40] National Highway Traffic Safety Administration. Traffic safety facts 2015 data: Pedestrians. Technical report, U.S. Department of Transportation, 2015.

[41] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach, 2018.

[42] Jianwei Niu, Jie Lu, Mingliang Xu, Pei Lv, and Xiaoke Zhao. Robust lane detection using two-stage feature extraction with curve fitting. *Pattern Recognition*, 59:225–233, 2016. Compositional Models and Structured Learning for Visual Recognition.

[43] Rachid Oucheikh, Mouhsene Fri, Fayçal Fedouaki, and Mustapha Hain. Deep real-time anomaly detection for connected autonomous vehicles. *Procedia Computer Science*, 177:456–461, 2020. The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020) / The 10th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2020) / Affiliated Workshops.

[44] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[45] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[46] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, 2007.

[47] Ravi Kumar Satzoda and Mohan Manubhai Trivedi. Drive analysis using vehicle dynamics and vision-based lane semantics. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):9 – 18, 2015. Cited by: 66.

[48] ASM Shihavuddin, Kabir Ahmed, Md Shirajum Munir, and Khandakar Rashed Ahmed. Road boundary detection by a remote vehicle using radon transform for path map generation of an unknown area. *International Journal of Computer Science and Network Security*, 8(8):64–69, 2008.

[49] GT Shrivakshan and Chandramouli Chandrasekar. A comparison of various edge detection techniques used in image processing. *International Journal of Computer Science Issues (IJCSI)*, 9(5):269, 2012.

[50] Sayanan Sivaraman and Mohan Manubhai Trivedi. Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):906–917, 2013.

[51] S Stephan, K Sarath, A Alen, and D Gamini. Efficient lane detection and tracking in urban environments. In *Proceedings of the 3rd European Conference on Mobile Robots*, 2007.

[52] Tsung-Ying Sun, Shang-Jeng Tsai, and V. Chan. Hsi color model based lane-marking detection. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1168–1172, 2006.

[53] Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021.

[54] Michael Tesfa. Lane detection for autonomous driving: Conventional and cnn approaches, 2021.

[55] Franco van Wyk, Yiyang Wang, Anahita Khojandi, and Neda Masoud. Real-time sensor anomaly detection and identification in automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1264–1276, 2020.

[56] Jung-Ming Wang, Yun-Chung Chung, Shyang-Lih Chang, and Sei-Wang Chen. Lane marks detection using steerable filters. In *Proc. of 16th IPPR Conf. on Computer Vision, Graphics and Image Processing*, pages 858–865, 2003.

[57] Yiyang Wang, Neda Masoud, and Anahita Khojandi. Anomaly detection in connected and automated vehicles using an augmented state formulation. In *2020 Forum on Integrated and Sustainable Transportation Systems (FISTS)*, pages 156–161, 2020.

[58] Yiyang Wang, Neda Masoud, and Anahita Khojandi. Real-time sensor anomaly detection and recovery in connected automated vehicle sensors. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1411–1421, 2021.

[59] Yue Wang, Dinggang Shen, and Eam Khwang Teoh. Lane detection using spline model. *Pattern Recognition Letters*, 21(8):677–689, 2000.

[60] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[61] World Health Organization. Road traffic injuries. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, 2021.

[62] Yang Xing, Chen Lv, Long Chen, Huaji Wang, Hong Wang, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. Advances in vision-based lane detection: Algorithms,

integration, assessment, and perspectives on acp-based parallel vision. *IEEE/CAA Journal of Automatica Sinica*, 5(3):645–661, 2018.

[63] Mucong Ye, Jingpeng Ouyang, Ge Chen, Jing Zhang, and Xiaogang Yu. Enhanced feature pyramid network for semantic segmentation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3209–3216, 2021.

[64] S. Yenikaya, G. Yenikaya, and E. Düven. Keeping the vehicle on the road: A survey on on-road lane detection systems. *ACM Computing Surveys*, 46(2):1–43, 2013.

[65] Youcheng Zhang, Zongqing Lu, Dongdong Ma, Jing-Hao Xue, and Qingmin Liao. Ripple-gan: Lane line detection with ripple lane line detection network and wasserstein gan. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1532–1542, 2021.

[66] Tu Zheng, Yifei Huang, Yang Liu, Wenjian Tang, Zheng Yang, Deng Cai, and Xiaofei He. Clrnet: Cross layer refinement network for lane detection, 2022.

[67] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE Transactions on Vehicular Technology*, 69(1):41–54, jan 2020.

APPENDIX

Algorithm 3 Initial

Input:

U : A matrix of pixels of a low resolution of the input image
maxPoolKernelSize : Kernel size for maxpool layer
maxPoolStride : Stride size for maxpool layer
IC : Number of Channels of Input Image
OC : Number of Channels of Output Image
convK : Kernel size for conv layer
convS : Stride size for conv layer
convP : Padding size for conv layer
Activation : PReLU Activation function

Output:

downsampledImg : Downsampled version of the Image

procedure
    $maxPoolOut \leftarrow$ MaxPooling(maxPoolKernelSize, maxPoolStride)(U)
    $strideOut \leftarrow$ Conv(IC, OC, convK, convP, convS)(I)
    $downsampledImg \leftarrow$ ConcatenateHorizontally($maxPoolOut, strideOut$)
    $downsampledImg \leftarrow$ batchNorm($downsampledImg$)
    $downsampledImg \leftarrow$ Activation($normImg$)
    return $downsampledImg$
end procedure

Algorithm 4 ConvBA

Input:

I : Input Feature Map

Algorithm 4 CONVBA Continued

Input:

IC : Number of input channels
OC : Number of output channels
S : Stride for convolution (default: 1)
kSize : Size of kernel for regular bottle neck
pad : Padding for regular bottle neck (default: 0)
dil : Space between kernel elements (default: 1)
bias : Specifies whether bias are in the convolutional layers (default: True)
Activation : Activation function to be used

Output:

downsampledImg : Downsampled version of the Image

1: **procedure**
2: $convOut \leftarrow$ Conv2d(IC, OC, kSize, S, pad, dil, bias)(I)
3: $bnOut \leftarrow$ BatchNorm2d($convOut$)
4: $aOut \leftarrow$ Activation($bnOut$)
5: **return** $aOut$
6: **end procedure**

Algorithm 5 DownsamplingBottleneck

Input:

I : Input Feature Map
R : Ratio which computes the number of channels after projection
C : Number of input channels
kSize : Size of kernel for regular bottle neck
pad : Padding for regular bottle neck
dropoutProb : Drop out probability
bias : Specifies whether bias are in the convolutional layers
activation : Activation function to be used
stride : Stride for convolution

Output:

downsampledFeatureMap : Downsampled version of the Feature Map

Algorithm 5 DownsamplingBottleneck Continued

procedure
    $OC \leftarrow$ C / R
    $maxOut \leftarrow$ MaxPool(kSize, stride)(I)
    $conv2Out \leftarrow$ ConvBA(C, OC, kSize+1, stride, bias)($maxOut$)
    $conv3Out \leftarrow$ ConvBA(C, OC, kSize-1, stride, bias)($conv2Out$)
    $dropOut \leftarrow$ Dropout(dropoutProb)($conv3Out$)
    $paddedMaxOut \leftarrow$ Zeropad(maxOut)
    $downsampledFeatureMap \leftarrow$ Add(paddedMaxOut, dropOut)
    return $downsampledFeatureMap$
end procedure

Algorithm 6 UpsamplingBottleneck

Input:
I : Input Feature Map
R : Ratio which computes the number of channels after projection
C : Number of input channels
kSize : Size of kernel for regular bottle neck
pad : Padding for regular bottle neck
dil : Space between kernel elements
asymmetric : Indicates if convolution is assymetric or not
dropoutProb : drop out probability
bias : Specifies whether bias are in the convolutional layers
activation : activation function to be used
stride : Stride for convolution (default: 1)
OutputSize : Size of output after unpooling and convtranspose
Output:
downsampledFeatureMap : Downsampled version of the Feature Map
procedure
    $OC \leftarrow$ C / R
    $mainConvOut \leftarrow$ ConvBA(C, OC, kSize-1, stride, bias)(I)
    $maxUnpoolOut \leftarrow$ MaxUnpool(kSize, unpoolOutputSize)($mainConvOut$)
    $conv1Out \leftarrow$ ConvBA(C, OC, kSize-1, bias)(I)
    $convTOut \leftarrow$ ConvTranspose2d(IC, IC, kSize, stride, bias)(I)
    $bnOut \leftarrow$ BatchNorm2d($convOut$)
    $aOut \leftarrow$ Activation($bnOut$)
    $conv3Out \leftarrow$ ConvBA(C, OC, kSize-1, stride, bias)($conv2Out$)

Algorithm 6 Upsampling Bottleneck Continued

10: $dropOut \leftarrow$ Dropout(p=dropoutProb)($conv3Out$)
11: $paddedMaxOut \leftarrow$ Zeropad(maxOut)
12: $downsampledFeatureMap \leftarrow$ Add(paddedMaxOut, dropOut)
13: **return** $downsampledFeatureMap$
14: **end procedure**

Algorithm 7 BottleNeck

Input:
I : Input Feature Map
R : Ratio which computes the number of channels after projection
C : Number of input channels
kSize : Size of kernel for regular bottle neck
pad : Padding for regular bottle neck
dil : Space between kernel elements
asymmetric : Indicates if convolution is assymetric or not
dropoutProb : drop out probability
bias : Specifies whether bias are in the convolutional layers
activation : activation function to be used
stride : Stride for convolution (default: 1)

1: **procedure**
2: $OC \leftarrow$ C / R
3: $conv1Out \leftarrow$ ConvBA(C, OC, kSize, stride, bias)(I)
4: **if** asymmetric = True **then**
5: $conv2Out \leftarrow$ ConvBA(C, C, (kSize,1), stride, (pad, 0), dilation, bias)($conv1Out$)
6: $conv2Out \leftarrow$ ConvBA(C, C, (1, kSize), stride, (0, pad), dil, bias)($conv2Out$)
7: **else**
8: $conv2Out \leftarrow$ ConvBA(C, C, kSize, stride, pad, bias)($conv1Out$)
9: **end if**
10: $conv3Out \leftarrow$ ConvBA(C, OC, kSize, stride, bias)($conv2Out$)
11: $dropOut \leftarrow$ Dropout(p=dropoutProb)($conv3Out$)
12: $addOut \leftarrow$ Add(I, conv3Out)
13: **return** $addOut$
14: **end procedure**

Table 1: ENet architecture

Name	Type
initial	
ENet Stage 1	
bottleneck1.0	downsampling
4× bottleneck1.x	
ENet Stage 2	
bottleneck2.0	downsampling
bottleneck2.1	
bottleneck2.2	dilated 2
bottleneck2.3	asymmetric 5
bottleneck2.4	dilated 4
bottleneck2.5	
bottleneck2.6	dilated 8
bottleneck2.7	asymmetric 5
bottleneck2.8	dilated 16

Algorithm 8 Binary Segmentation Branch

Input:

I : Input Feature Map
stageFourOC : Number of output channels in stage four of ENet
stageFourIC : Number of input channels in stage four of ENet
stageFourIC : Number of input channels in stage five of ENet
kSize : Size of kernel for regular bottle neck
pad : Padding for regular bottle neck
dp : drop out probability
bias : Specifies whether bias are in the convolutional layers
act : activation function to be used
S : Stride for convolution
binarySegDim : The output dimension of binary segmentation

Output:

bOut : Output of binary segmentation

Algorithm 8 Binary Segmentation Branch Continued

procedure
    $upOut \leftarrow$ UpsamplingBottleneck(stageFourOC, stageFourIC, dp, act)(I)
    $regularOut \leftarrow$ RegularBottleneck(stageFourIC, pad, dp, act)($upOut$)
    $regOut \leftarrow$ RegularBottleneck(stageFourIC, pad, dp, act)($regularOut$)
    $upOut \leftarrow$ UpsamplingBottleneck(stageFiveOC, stageFiveIC, dp, act)($regOut$)
    $regOut \leftarrow$ RegularBottleneck(stageFiveIC, pad, dp, activation)($upOut$)
    $bOut \leftarrow$ ConvTranspose(stageFiveIC, binarySegDim, kSize, S, pad, bias)($regOut$)
    return $bOut$
end procedure

Algorithm 9 Embedding Branch

Input:
I : Input Feature Map
stageFourOC : Number of output channels in stage four of ENet
stageFourIC : Number of input channels in stage four of ENet
stageFourIC : Number of input channels in stage five of ENet
kSize : Size of kernel for regular bottle neck
pad : Padding for regular bottle neck
dp : drop out probability
bias : Specifies whether bias are in the convolutional layers
act : activation function to be used
S : Stride for convolution
embedDim : The output dimension of the embedding branch

Output:
binaryOut : Output of binary segmentation

procedure
    $upsampleOut \leftarrow$ UpsamplingBottleneck(stageFourOC, stageFourIC, dp, act)(I)
    $regularOut \leftarrow$ RegularBottleneck(stageFourIC, pad, dp, act)($upsampleOut$)
    $regOut \leftarrow$ RegularBottleneck(stageFourIC, pad, dp, act)($regularOut$)
    $upsampleOut \leftarrow$ UpsamplingBottleneck(stageFiveOC, stageFiveIC, dp, act)($regOut$)
    $regOut \leftarrow$ RegularBottleneck(stageFiveIC, pad, dp, activation)($regOut$)
    $embedOut \leftarrow$ ConvTranspose(stageFiveIC, embedDim, kSize, S, pad, bias)($embedOut$)
    return $embedOut$
end procedure

Algorithm 10 PerspectiveTransformation

Input:

perspectiveParam: Perspective transformation parameters for the image

P : Lane pixels such that $p_i = [x_i, y_i, 1]^T \in P$

Output:

P': Projected lane pixels

procedure

    $H \leftarrow$ MakePerspectiveMatrix($perspectiveParam$) // Algorithm 12

    $P' \leftarrow$ HP

    return P'

end procedure

Algorithm 11 Lane Fitting

Input:

E : Lane instance and their associated lane pixels

Output:

V: parameters of a spline for each lane instance

procedure

    $splines = []$

    for lane instance $\in$ E **do**

        lane instance pixels $\leftarrow$ PerspectiveTransformation(pixels of lane instance) // Algorithm 10

        X $\leftarrow$ x-coordinates of lane instance pixels in the form $[x_1, x_2, \ldots, x_n]$

        Y $\leftarrow$ y-coordinates of lane pixels in the form $[y_1, y_2, \ldots, y_n]$

        $w \leftarrow (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}'$

        append w to V

    end for

    return V

end procedure

Algorithm 12 MakePerspectiveMatrix

Input:

A: Encoded version of the input image
inChannels : Number of Channels of Input Image
outChannels: Number of Channels of Output Image
filters : Array of filters to be used in each iteration
kernelSizes : Array of filter sizes to be used in each iteration
maxPoolKernelSize : Kernel size of maxpool to be used
maxPoolStride : Stride for maxpool layer
activation : Activation function to be used
N : Number of times to loop

Output:

perspectiveParam : Perspective transformation parameters for the image

1: **procedure**
2: **for** $i \leftarrow 1$ to N **do**
3: $convOut \leftarrow$ Conv(inChannels, internalChannels, filters[i], kernelSizes[i])(I)
4: $bnOut \leftarrow$ BatchNorm2D(inChannels)($convOut$)
5: $activationOut \leftarrow$ Activation(activation)($bnOut$)
6: $convOut \leftarrow$ Conv(inChannels, internalChannels, filters[i], kernelSizes[i])(I)
7: $bnOut \leftarrow$ BatchNorm2D(inChannels)($convOut$)
8: $activationOut \leftarrow$ Activation(activation)($bnOut$)
9: $maxPoolOut \leftarrow$ MaxPooling(maxPoolKernelSize,maxPoolStride)($activationOut$)
10: **end for**
11: $perspectiveParam \leftarrow$ Linear($maxPoolOut$)
12: **return** $perspectiveParam$
13: **end procedure**