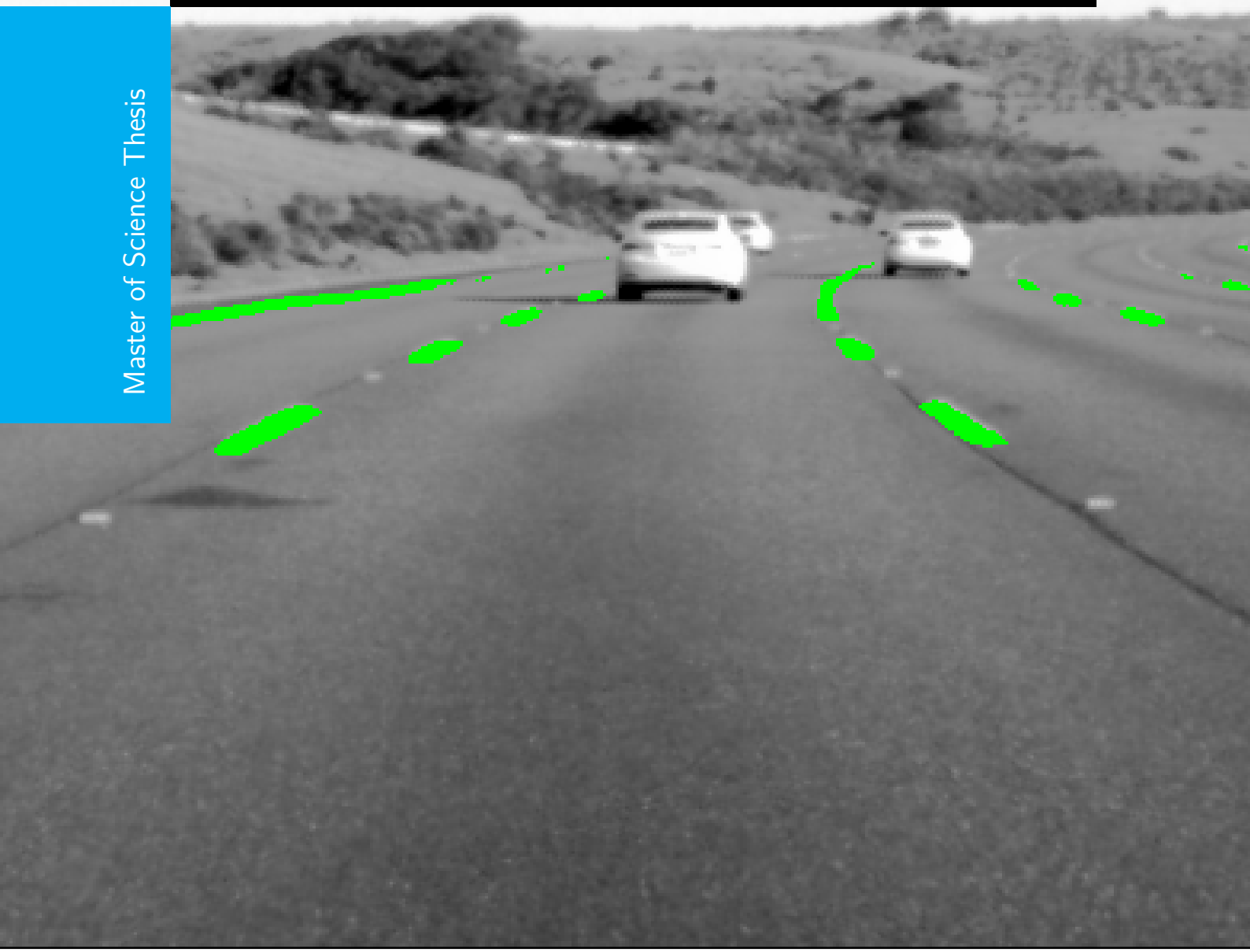
Lane Detection using Spatio-Temporal Attention

Sandeep Patil

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Lane Detection using Spatio-Temporal Attention

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

Sandeep Patil

August 31, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Copyright © Cognitive Robotics (CoR)
All rights reserved.

Delft University of Technology
Department of
Cognitive Robotics (CoR)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Lane Detection using Spatio-Temporal Attention

by

Sandeep Patil

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering

Dated: August 31, 2021

Supervisor(s):

prof.dr.ir.H.Hellendoorn

Dr.ir.H.Farah

Ir.Y.Dong

Reader(s):

Abstract

Lane detection represents a fundamental task for automated/autonomous vehicles. Current lane detection methods do not provide the versatility of real-time performance, robustness, and accuracy required for real-world scenarios. The reasons include lack of computing power while being portable and inability to observe the continuity and structure of lane lines over a sequence of images.

An investigation into the present methods in the literature reveals that deep learning networks cannot focus on relevant images and critical parts of the images. The neural networks implemented with max-pooling operations can cause a loss of information at a granular level of the image during the downsampling of images. Obtaining a fixed set of lane locations will restrict the number of lane lines detected. It hinders the generalisability of the network. This thesis aims to introduce a novel spatio temporal method that can focus on lane lines and key features to increase the robustness and accuracy of lane line detection.

The spatio temporal attention network based on Long Short Term Memory (LSTM) units tested on the tvtLane dataset provided an accuracy of 98.1443%, the precision of 0.8873, and F1-score of 0.9108. The precision and F1-measure are the highest when compared to state-of-the-art lane detection networks. The spatio temporal FC attention network produces better accuracy and precision on TuSimple dataset than state-of-the-art networks with 98.2078% and 0.8861, respectively. Testing on the LLAMAS dataset, the network achieved an average precision of 0.8028 and corner recall of 0.7183. The results show the robustness and high accuracy on two different datasets with unique distributions. Although the network is trained on datasets with a maximum of five lanes, it can detect more than five lanes in an image. The network is also able to detect lanes on the unseen Netherlands dataset.

Table of Contents

List of Figures

List of Tables

Acknowledgements

The past two years has been an enlightening experience for me. I had the opportunity to expand my knowledge, skills, and network in leaps and bounds. I have had memorable days at the TU and in Delft. This was possible because of:

1. My supervisor Prof.dr.ir.Hans Hellendoorn for his invaluable guidance in my thesis and always supporting me. He always made sure that I am progressing in the right direction.
2. My supervisor Dr.ir.Haneen Farah who believed in me and gave me the opportunity to do the thesis.
3. My co-supervisor Ir.Yongqi Dong for his encouragement, immense help, and discussion about new possibilities in my thesis.
4. Dr.ir.Julian Kooij who agreed to be a part of my thesis committee.
5. My parents' constant support in my endeavours and making sure I am always fit and healthy.
6. My friends Dheeraj, Akshay, Zeeshan, and Sushant back home who listened to all my complaints and worries and believed in me.
7. My gang Ayushi, Anjali, Chinmay, Punith, Shishir, and Viswanath at the TU who always had my back during the two years.
8. ASIMOV study association members Emilio, Mees, Luuk, Kirsten, and Marie who with their shenanigans and stories made my final year lively.
9. Asst. MSc Coordinator Karin who supported me during my recovery after the accident.

Sandeep Patil
Delft, University of Technology
August 31, 2021

Chapter 1

Introduction

According to European Road Safety Observatory (ERSO) 23.926 fatal accidents occurred in EU countries in 2016, and 47% of these accidents involve a car or taxi when compared with bicycles, motorcycles, moped, lorries, and buses [38]. One of the leading causes of these accidents is human error. Modern vehicles are fitted with an array of sensors that aid in human driving, taking control in an emergency, or driving autonomously.

Taxonomy and definitions proposed by the Society of Automotive Engineers (SAE) [19] suggests a division of levels of automation from no automation (SAE level 0) to full automation (SAE level 5). Figure 1-1 shows general definitions and examples of tasks in each automation level. Lane detection is a fundamental task for all vehicles ranging from SAE level 0 to SAE level 5. Lane departure warning system in SAE level 0 vehicles, Lane centering system in SAE level 1 and 2 vehicles use the detected lanes. These systems constitute automated driving systems. In SAE level 3 to level 5 vehicles, environment perception remains an essential task, and lane detection is one of the building blocks [42]. These systems constitute autonomous driving systems.

These vehicles (SAE level 1 and SAE level 5) share the same road as human drivers. Presently, lane detection primarily relies on visual sensors [8]. Hence, **lane detection** remains a task of detecting the line of points on the road visually, which determines an effective driving direction for all vehicles.

Lanes are detected through sensors like mono-camera, stereo-camera, radar, and lidar. The mono-camera sensors can detect lane lines, objects such as road signs and pedestrians. The stereo camera sensors can detect curbs, slopes, or a 3D model of the road with sufficient accuracy. Radars can detect the regions based on their reflectivity and detect vehicles and other objects. Lidar can estimate the roughness of the road surface and detect road edges due to the change in the 3D surface from the road to curbs [5].

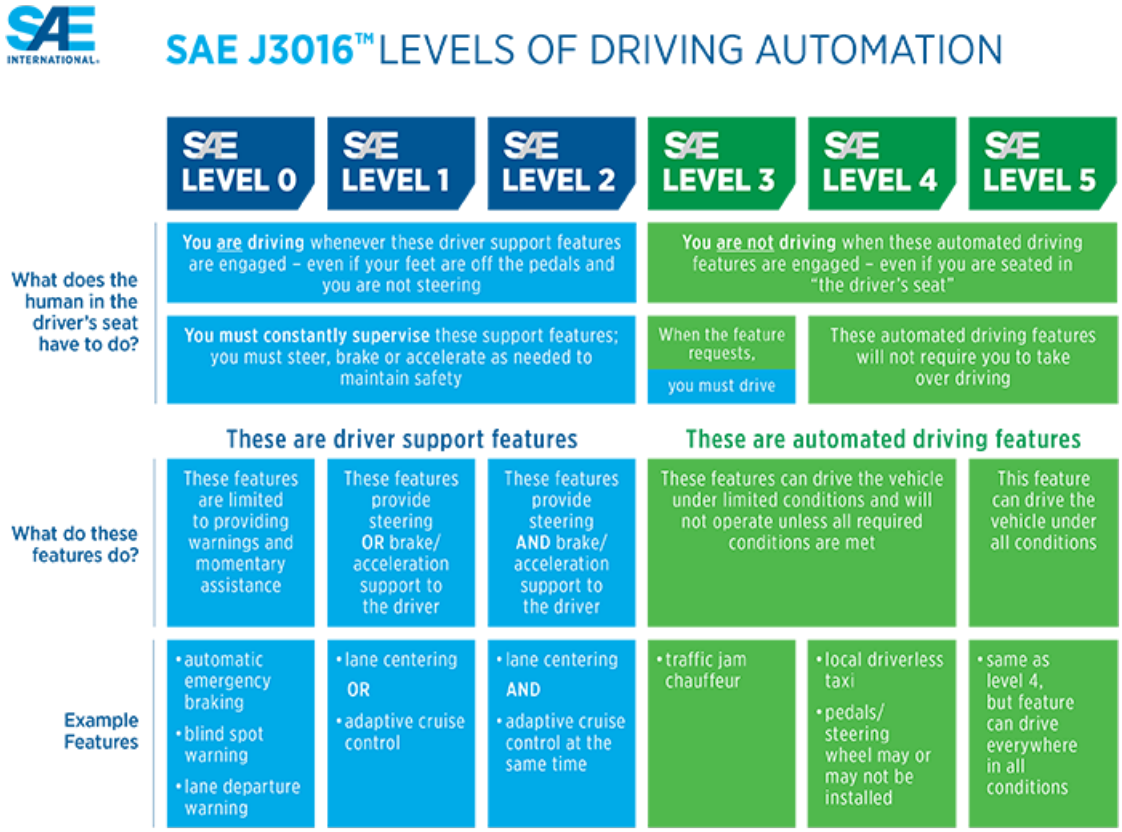


Figure 1-1: SAE J3016: Levels of Driving Automation

Vision sensors, which include a mono-camera and stereo camera, are the most successful due to the amount of information they provide [31]. They constitute passive sensors that do not emit any signals and hence do not interfere with each other at close proximities. Active sensors, which include radar and lidar, tend to have lower resolution and slower scanning speeds. Active sensors tend to interfere with each other at close proximities due to active signal propagation. Active sensors cost more than passive sensors. Hence, considering the information gain, speed of capturing data, and cost, a mono-camera sensor is widely used for lane detection.

Lane detection in itself is a straightforward task in which filters can detect the lane lines, but there are many cases in which the lane lines are not visible [5]. Figure 1-2 visualizes these cases, which include snow, rain, night conditions, shadows, and severe occlusions. The false-positive lane detection can affect the control of vehicles undesirably. In cases like the Lane Departure Warning (LDW) systems of non-autonomous vehicles, many false positives annoy the driver and may cause the system to fail. Hence, lane detection is a nontrivial task for automated/autonomous systems of vehicles, and finding a solution for the lane detection task which overcomes the challenges mentioned above is of utmost importance.

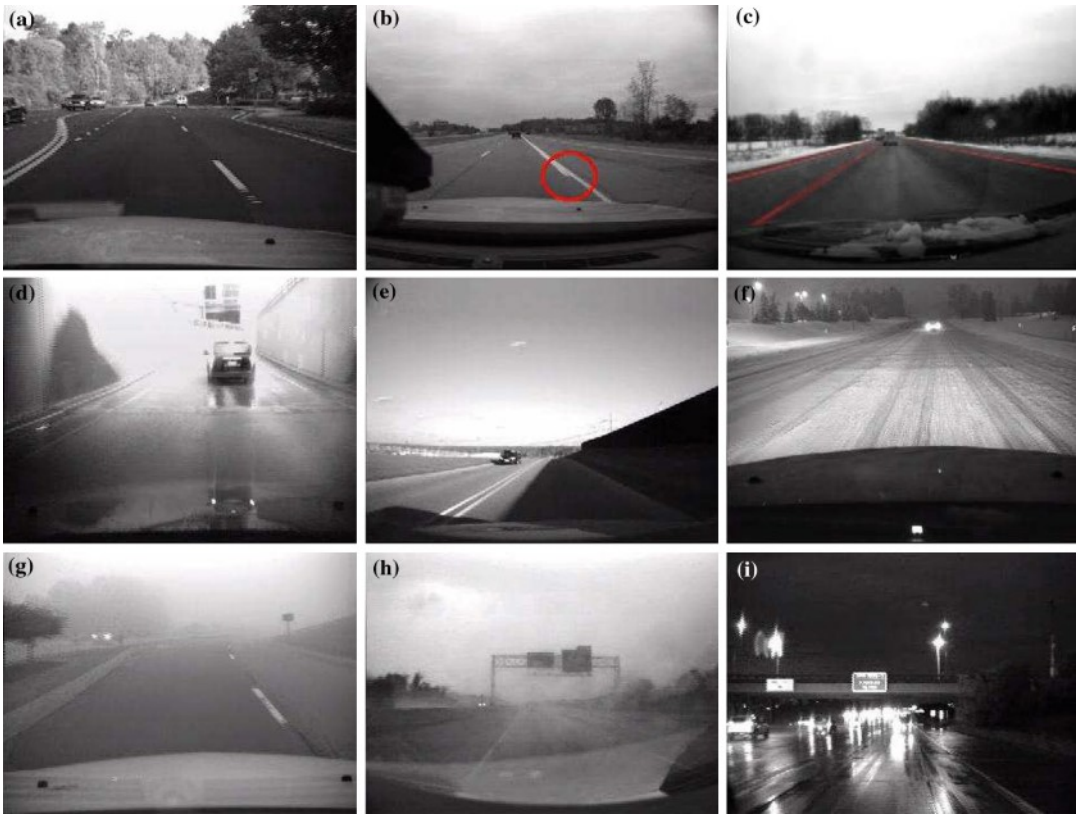


Figure 1-2: Challenges experienced during lane detection. (a)different lane shapes (b) varying lane thickness (c) lane width variation (d) over-exposure in the image (e) shadow covering the road (f) snow-covered road (g)low-visibility (fog) (h)low-visibility (rain) (i) wet and reflective road ([5])

1-1 Problem Description

A Lane needs to be perceived in all conditions and environments to place the vehicle centered on the road and during the lane change operation. There are adverse conditions like over-exposure (figure 1-2d), shadows (figure 1-2f), worn-out lane markings, occlusion due to vehicles in the front, and wet and reflective road (figure 1-2i). Consider an autonomous vehicle driving through a tunnel. As it exits the tunnel, there is an abrupt change in brightness detected by the onboard camera. During this time, the camera will not detect the lanes to place the vehicle centered on the road. In another case, where a large vehicle is obstructing the view of the road ahead, the lane lines are partially or completely occluded, and lane detection becomes a challenging task. Moreover, detecting false lanes can change the course of the vehicle and put the passengers at risk. When lane detection fails, the vehicle can veer off the road or cause accidents. How to overcome these challenges? How to reduce these false positive rates to a minimum?

Much research has been conducted in this direction, and many methods have been suggested, which are discussed briefly by Liu et al. and BarHillel et al. [26] [5]. Liu et al. found deep learning methods to be promising and that it can be further developed [26].

To summarize, the current methods such as LaneNet, CNN-LSTM(SegNet+, UNet+), CNN-GRU(UNet+) are not capable of selectively focusing on images in a sequence and ocassionally misclassify the lane lines [32][48][47].

Research Questions

Following the identified research gaps, two research questions are defined:

1. How to extract spatial-temporal information and capture relevant information over time?
2. How to reduce false positive lane detection?

1-2 Contributions

The primary contribution of this thesis is a novel sequence-based deep learning model which accepts image sequences as input and produces pixel-wise segmented lane outputs.

In particular 1)Implementation of the Spatio-temporal attention model to focus on relevant images intuitively. 2)Use the convolutional neural layer as a replacement for max-pooling layer to extract low-level features more efficiently. 3)Evaluation of the model on various lane datasets for robustness.

1-3 Outline

In this thesis report, related works on existing lane detection methods are discussed in Chapter 2. Implementation method, network training choices are discussed in Chapter 3. Experiments and results are elaborated in Chapter 4. The results of the experiments are discussed in detail. Conclusions are drawn, and recommendations for future work are presented in Chapter 5.

Chapter 2

Related work

Following the importance of lane detection for vehicles from the Introduction chapter, the main goal is to detect lane lines consistently, coherently, under any given weather conditions and driving scenarios. Relevant methods to solve the challenging task of lane detection are discussed in this chapter.

The lane detection algorithms can be broadly classified into traditional and deep learning methods. Traditional lane detection methods involve manual mapping functions or algorithms which define the relation between observed road features and lane lines. Deep learning methods do not require designing a mapping function between the observed road features and lane lines. Instead, the methods learn the mapping function by processing large datasets with varied features.

Another distinction of methods in both traditional and deep learning methods can be based on the input. Spatial methods process only a single image, whereas spatio-temporal methods process a sequence of images and use the temporal information between the images in sequential time steps. An overview of the methods discussed in traditional lane detection methods and deep learning methods is visualized in Figure 2-1.

In traditional lane detection (figure 2-1), a general methodology consists of image preprocessing to denoise and remove the camera perspective, detection of relevant lane points, clustering the lane points, and fitting a curve to produce the lane lines. A postprocessing step is also considered to reduce false positive lane detection. Temporal data is considered in some methods and is integrated in the process after relevant lane point detection. These steps are explained in detail in the following sections.

In deep learning methods (figure 2-1), convolutional neural networks preprocess the image and either produce semantic segmentation (pixel-wise classification) or predict the location

of lane points through downsizing or downsampling the images. These outputs are postprocessed to remove outliers. Recurrent neural network-based methods follow similar steps of preprocessing, downsampling and extend the method to include the temporal information from previous images through Long Short Term Memory (LSTM) units or Gated Recurrent (GRU) units. The images are upsized or upsampled to their original size to detect lanes. Generative adversarial neural networks include a generator which generates a detected lane image and learns via discriminating between a generated image and an original image. These methods are explained in detail in the upcoming sections.

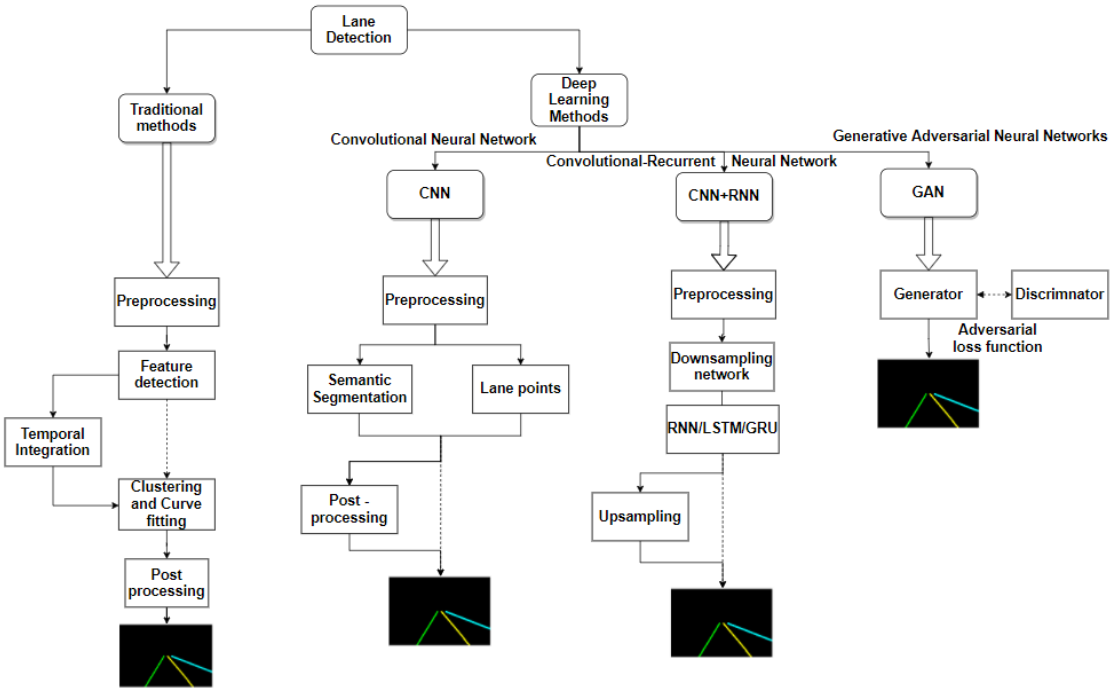


Figure 2-1: Overview of Traditional lane detection method and deep learning method of lane detection

2-1 Traditional Methods of Lane Detection

Traditional lane detection methods use a heuristic approach to detect and model the lanes. Mono-camera sensor outputs a sequence of images of the road and matrix operations can be applied to these images. Using filters and operations, the lane points can be detected and grouped in lane line instances. The generalized system consists of image preprocessing, feature detection, temporal information integration, clustering, curve fitting, and postprocessing. These steps are visualized in Figure 2-2 below.

The algorithms studied in the literature are a subset of the generalized system and do not use all the steps mentioned above such as the method suggested by Low et al. which does not use temporal data from the image sequences, but follows the other steps mentioned in Figure

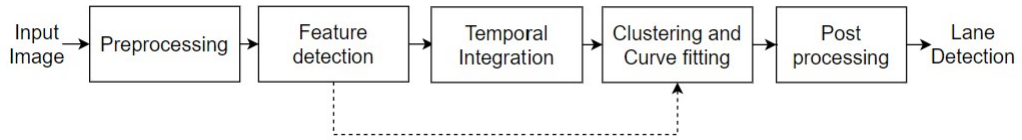


Figure 2-2: Outline of traditional method for lane detection task

2-1 [28]. Hence, each step in the system is discussed in detail, and a comparison between different algorithms is provided.

2-1-1 Image preprocessing

The lane lines are orange, white, or yellow. One method to detect the lane lines is by binarizing and thresholding the images. However, McDonald et al. inferred that thresholded images suffered due to the glare on the road surface [31]. Another method to detect the lane lines is by using the Sobel edge detector. However, in this case, road barriers are also detected along the lane lines as they are parallel to the road. Assuming glare to be occurring only on the road surface, to avoid detecting the road barriers and detect lane lines precisely, thresholding and Sobel edge detection operation can be combined using 'AND' operation. An example of this operation is visualized in Figure 2-3.

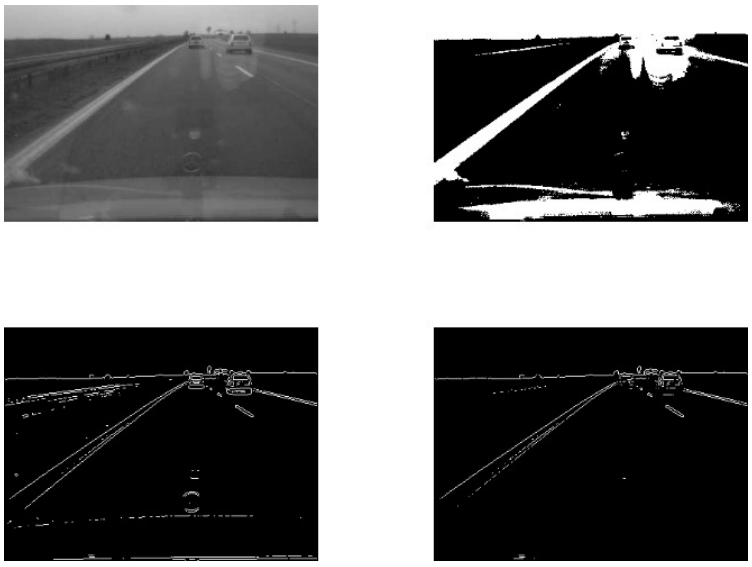


Figure 2-3: Image pre-processing in the case of McDonald et, al. [31]

Due to the camera sensor's perspective effect, which is mounted at an angle to the vehicle, the lane lines tend to intersect at the horizon. To avoid this effect, Inverse Perspective Mapping (IPM) is used [1]. Through this, the vehicle can be assumed to be parallel to the lanes all time. During the postprocessing, the IPM can be reversed to obtain the original image. An example of IPM is shown in Figure 2-4.

Figure 2-4: Inverse perspective mapping [1]

The image obtained from the sensor is often noisy and large. To remove the noise from the images, topology-based methods such as dilation and erosion [28], gaussian filters can be used. 2D-gaussian filters are applied in the case of Aly et al., which is based on the lane dimensions to denoise the image [1]. Other operations such as cropping, grey scaling can also be applied to reduce the size of the input image and noise. These operations are visualized in Figures 2-5 and 2-6.

Figure 2-5: Pre-processing techniques such as cropping and greyscale [28]

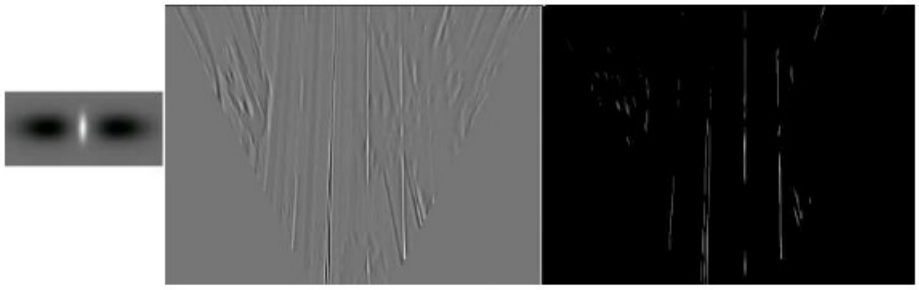


Figure 2-6: Gaussian filter used to denoise the image [1]

2-1-2 Feature Detection

The lane lines form ridge-like structures due to the change in intensity between the lanes and the background when binarized. Hence, edge detectors such as Canny edge detector and Hough transform can be used. A Canny edge detector uses filters to denoise the images and detects the gradients in the image. The gradients are then combined using a weighted average and then denoised again. An example of canny edge detection can be observed in Figure 2-7.

Figure 2-7: Before(left) and after(right) applying canny edge detection

Aly et al. uses the simplified Hough transform to count the number of lines [1]. Hough transform is a method to transform a set of points in the image plane to a *Hough space* (parameter space) [31]. The parameters are the radius ρ from the origin, and the angle θ with the horizontal axis. Two or more points having the same (ρ,θ) values are considered to be a part of the same straight line. Lane lines are considered if there are many concurrent curves in the *Hough space*. McDonald et al. defines a window of theta values at the horizon where the lane lines appear to intersect[31]. The values of ρ and θ in the Hough space are considered parameters of lane lines, and other values in the Hough space are rejected as outliers. An example of an Hough input image and the corresponding output in the Hough space is shown in Figure 2-8. Aly et al. suggested using the Artificial Neural Network to generate the possible lane points[1].

2-1-3 Temporal information

McDonald et al. stated the assumption that the intersection of lane lines at the horizon might not always hold, especially during occlusion in the image and proposed the usage of temporal information of the measured values to predict future values [31]. The temporal information is generated using the exponential averaging process as shown in Equation 2-1 below:

$$\tau_i = \alpha * x_i + (1 - \alpha) * \tau_{(i-1)} \tag{2-1}$$

Here τ is the predicted value, x is the measured value, and α is the weight parameter to influence the measured value and the predicted value. The ego motion of the vehicle determines the α value. In Equation 2-1, the predicted value tends to deviate from the true value based

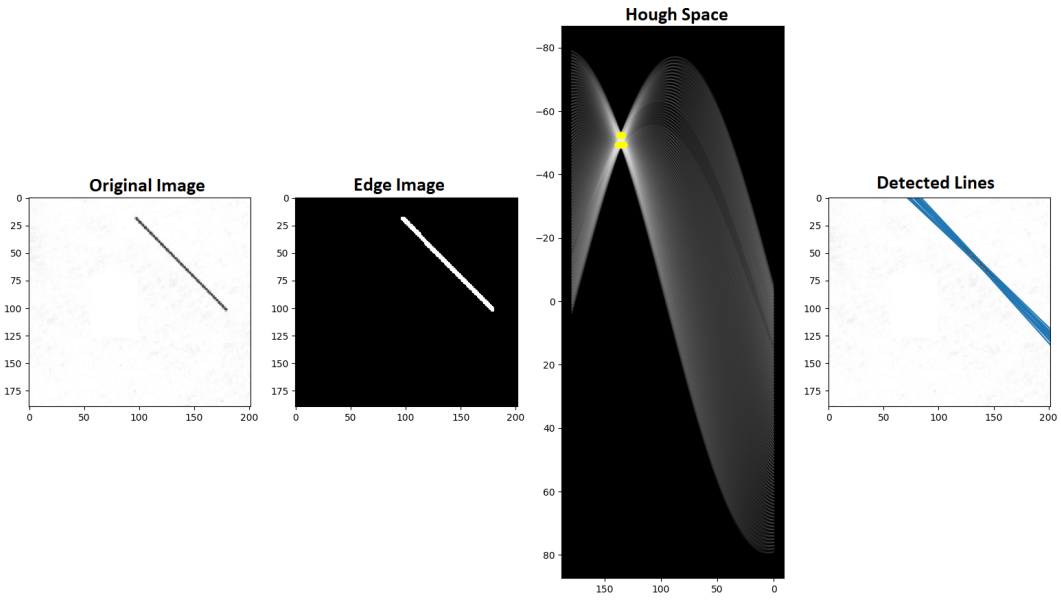


Figure 2-8: An example of hough input image and the corresponding output in the hough space

on the prior value. To avoid this, a threshold is enforced on the predicted value, and a new predicted value is considered only when it is largely different from the measured value.

Another method proposed by Kim et al. uses particle filtering to enforce the temporal information of lane detection—the vehicle motion, which is modeled as a Gaussian distribution for simplicity [21]. The lane boundary hypothesis is generated based on the previously detected control points of the lane lines. The control points are updated with the motion of the vehicle. The final position of control points is estimated by the weighted sum of the scores of hypotheses. When a lane pixel is detected in the current frame, the lane line control points are slightly adjusted.

2-1-4 Clustering and Curve fitting

After a set of plausible lane lines are obtained through Hough transform or canny edge detectors, Low et al. group the lane lines into left and right lanes and average them to produce a single left and right lane [28]. When the lane lines are obtained as a set of points they are grouped into line segments [21]. A hypothesis is generated from a random set of line segments. Cubic spline-based RANdom SAmple Consensus (RANSAC) method combines the line segments into one lane line. A cubic spline has five control points, and three control points are found with this method. The remaining two control points are obtained by extending the lane line to the bottom of the image and considering the last selected pixel within the upper bound. A curve score is generated to verify the line's fit and an overlapping test with the original line segment.

$$\text{CurveScore} = (1 - \lambda)\sum_{m} 1 \tag{2-2}$$

Where λ is the penalty for minimum description length, m is the supporting lane marking pixel on the curve. Kim et al. implemented $\lambda = 0.1$ for hypothesis generated from two sets of line segments and $\lambda = 0.2$ for the hypothesis generated by three sets of line segments [20]. The set of lane pixels detected and generated hypotheses is visualized in Figure 2-9 below.

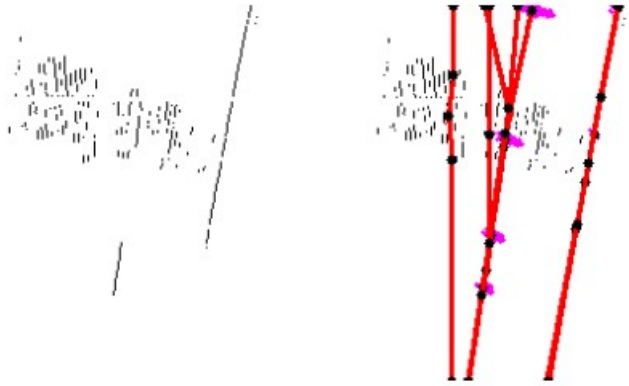


Figure 2-9: Lane pixels detected and generated hypothesis [20]

2-1-5 Post processing

In the case of Kim et al., there are two hypotheses generated, one based on the detected lane control points in the current frame and the other based on the hypothesis of lane control points generated from the particle filtering method [20]. These two are combined using probabilistic reasoning for decision making. A dynamic Bayesian network-style formulation is used, where $e = (e_c, e_t, e_p)$ is the evidence collected. e_c is the collective evidence, e_t is the transitional evidence obtained from the temporal information, and e_p is the past evidence collected from the previous frames. The probability of existence of the lane pixel x is calculated using the Bayes rule as:

$$P(x|e_c, e_t, e_p) = \frac{P(e_c|x)P(x|e_t, e_p)}{P(e_c)}, \tag{2-3}$$

Another method to track the detected lane lines is by using Kalman filters. Borkar et al. uses the parameters ρ and θ to fit the lines on detected lane points [7]. The parameters, ρ, θ, $\dot{\rho}$, and $\dot{\theta}$ define the state vector x and observation vector y, where $\dot{\rho}$ and $\dot{\theta}$ are derivatives of ρ and θ estimated using the previous and the current frame. The process is assumed to be uncorrelated, and noise in the state and measurement models is white. The covariance matrices for this system are constant and diagonal. The state transition matrix is given as:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The observation matrix C is the identity matrix. When the lane marker is not detected, matrix C is set to zero, and the lanes are purely based on the Kalman filter's prediction.

2-2 CNN based methods

Neural networks based on a series of convolution operations constitute Convolutional Neural Networks (CNN). Convolution operations have a special property of extracting complex shapes. These operations can be used to extract lane lines from images. A general methodology includes preprocessing, semantic segmentation or rowwise classification and postprocessing. These steps are explained in further sections.

2-2-1 Pre-Processing

Unlike traditional methods, a Convolutional Neural Network (CNN) does not require a large amount of preprocessing. There are generally four tasks in this phase:

1. Image resize
2. Inverse Perspective Mapping
3. Cropping
4. Grey-scaling

To reduce the network's training time and size, the images are resized to lower resolutions, such as TuSimple dataset images of size 1280x720 are resized to 256x512 [46]. Other networks such as Ko et al. have dedicated resizing layers that can extract relevant features through series of convolutions and max-pooling layers [23]. The pre-processing techniques include cropping of image, grey-scaling, and inverse perspective mapping (IPM) [25]. As the cameras are mounted in the moving vehicle, they have a certain angle of view which renders the lines distorted due to the perspective. Inverse perspective mapping operation in the two-dimensional domain is carried out to remove this perspective effect. This will distribute the features homogeneously among the pixels.

From the resized image, the task is to extract the features relevant for lane detection. This can be done in two main ways: semantic segmentation method and rowwise classification method. These methods vary in terms of computation memory requirements, speed, efficiency, and generalization of the network to all types of road conditions.

2-2-2 Semantic Segmentation methods

The Semantic segmentation task is a part of computer vision tasks where each pixel in an image is labeled by its class. This allows localization and boundary detection of each object in the image. For an autonomous vehicle, understanding different objects such as buses, bicycle, pedestrians, and signboards on the road provides more contextual information [11]. Lane detection is part of contextual learning. Lanes are inherently curved lines detected by binary classification of each pixel as $\{\text{lane}, \text{not-lane}\}$.

The semantic segmentation task is usually carried out through an encoder and decoder network [18]. Here the encoder network downsamples the image input to obtain a meaningful representation, and the decoder network upsamples the representation from the encoder and labels the pixels with its corresponding class. In the case of Hou et al., the author presents a self-attention distillation module to the semantic segmentation task, where after each encoding block, a layer-based activation map, known as **attention map** is implemented [18]. The attention map transfers important features such as edges, shapes from previous layers to the next layer which is generated by the attention generator. There are four encoding blocks and two decoding blocks, and a small binary classification network predicts the lane's existence. The network is visualized in Figure 2-10:

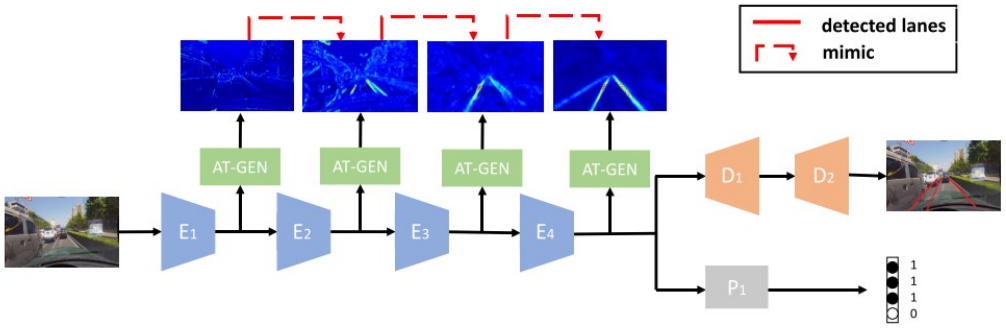


Figure 2-10: Example of semantic segmentation method based networks [18]

The attention map generated is propagated to the next layer to reinforce the knowledge obtained in the preceding layer. This method provides great accuracy along with efficiency while having a lightweight backbone.

The Instance segmentation method is similar to the semantic segmentation method, but it also detects the objects and the number of instances of their appearance. In the case of Ko et al. , an hourglass style network is used to extract the features [23]. An hourglass style network is an encoder-decoder style network with skip connections between equal-sized upsampling and downsampling layers. The hourglass style network outputs are confidence, offset, and feature, and the confidence output is forwarded to the next block of the network. This network is visualized in Figure 2-11.

The confidence output gives the confidence of the lane point existing in the pixel grid. The offset output gives the exact location of each point. The feature output groups the detected

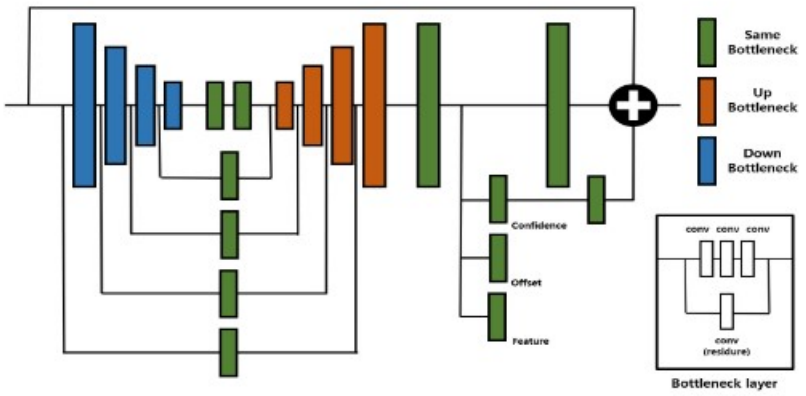


Figure 2-11: hourglass style network [23]

pixels into instances. These outputs are obtained by passing the last layer of the hourglass network through a series of filters, 1, 2, and 4 filters for confidence, offset, and feature, respectively. There are two hourglass networks in the series, and the network is visualized in Figure 2-12.

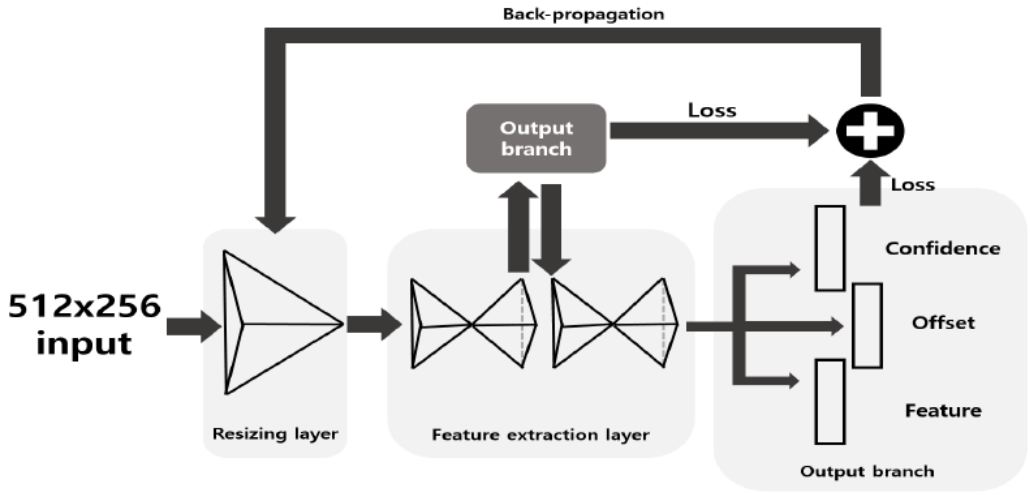


Figure 2-12: Example Architecture of Instance segmentation method [23]

The outputs do not give complete information, hence they are postprocessed to obtain the relevant lane lines in the image. This is discussed in section 2-2-4.

2-2-3 Row-wise classification methods

Row-wise classification methods take advantage of the innate shape of the lanes. The network predicts lanes' existence in any given row of the image and returns the lane vertices' position along with it. The lane lines can be characterized by long and thin lines. Squeezing the network horizontally (reducing the number of rows through down-sampling) can represent

the lane line accurately.

Yoo et al. discuss the effectiveness of this method and propose an End-to-End Lane Marker Detection network (**E2E-LMD**), which is a two stage network where the image representation is learned via the encoder-decoder network and the spatial resolution is halved to reduce the computational requirements in the first stage [46]. The second stage comprises shared reduction layers consisting of a novel horizontal reduction module (**HRM**). The horizontal reduction module comprises skip connections, including average pooling layers followed by $[1 \times 1]$ convolutions. This is followed by rearranging the elements of the layer with input tensor (channel $*$ height $*$ width) to (r $*$ channel $*$ height $*$ width/r), where r denotes the pooling ratio, which moves the spatial information to the channels and effectively compresses the columns of the input image matrix to an individual lane-wise representation.

Yoo et al. concluded that the number of HRMs should be equal to the number of lanes to get accurate results [46]. The network architecture and the Horizontal Reduction Module are shown in Figure 2-13.

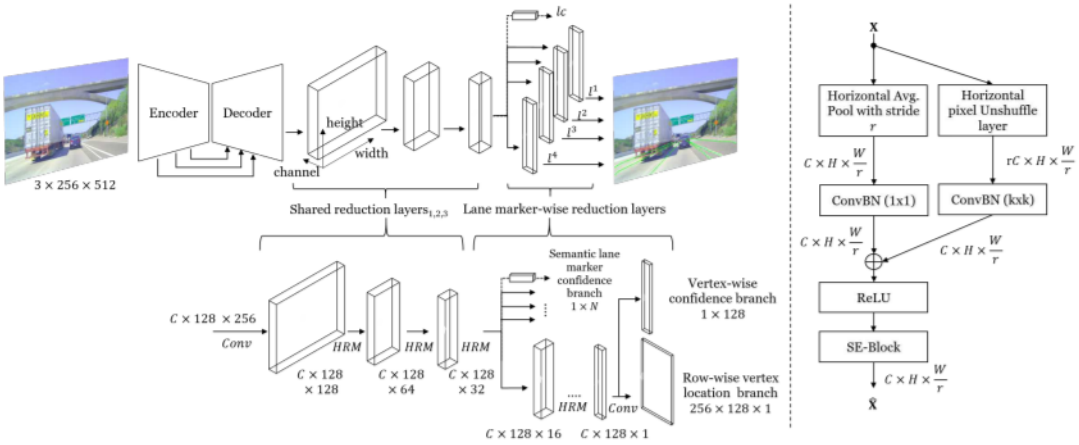


Figure 2-13: Example architecture of row wise classification method and HRM [46]

Although this is an end-to-end method of lane detection, it cannot adapt to an increase in the number of lanes, hence cannot generalize to real-world scenarios.

Another method by Qin et al. provides a solution for lane detection via row-anchor-wise classification. Instead of classifying the image, which considers a large number of pixels based on the resolution, the image is divided into cells of much larger size than a pixel, and the probability values of finding the lane in the cells are calculated [36]. Hence the goal is to select lane "anchors" at predefined rows of the image using global features instead of segmenting every lanes' pixel based on a local receptive field. The lanes are then detected in the cell matrix's rows and are combined to form a smooth lane.

It also uses prior information of lane lines such as smoothness of lanes, rigidity to solve occlusion. The number of calculations through the model suggested by Qin et al. decreased by 10^2

fold in comparison to segmentation methods [36]. In Figure 2-14 below, Res blocks contain the proposed network's backbone, which is either ResNet18 or ResNet34 [15]. This backbone contains a set of CNN layers to downsample and extract features. The Group classification block provides classification-based prediction. The final layer of group classification is of the size of the original image and it is divided into rows, also known as row anchors. In each row anchor, the lane points existences are detected and classified into their respective classes. The auxiliary segmentation network performs loss calculation on the global context and local features.

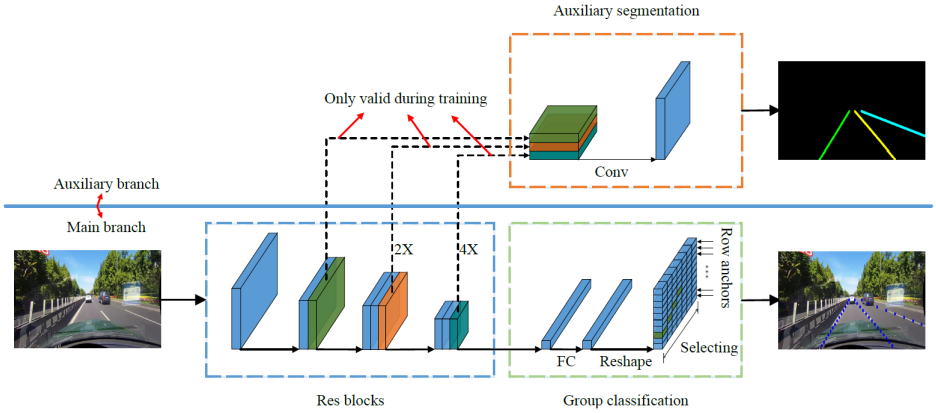


Figure 2-14: Overall architecture of Ultra Fast Structure aware Deep Lane Detection [36]

Qin et al. assume that the majority of the curve lanes are straight due to the perspective effect, which might not be the case always [36]. Hence this method may not be able to generalize to all driving scenes.

2-2-4 Post-processing

Post-processing is carried out to remove outliers and draw a structured lane line. Sometimes more than one lane line which overlaps each other could be detected by the network. In these cases, Ko et al. suggest selecting lane points based on the proximity, their proximity to the center of the image, and the respective lane line they belong to {left lane, right lane} [23]. Six points are chosen as 'starting points' and are divided into lower three points and upper three points. If the upper three points lie on the left side of the image, then the leftmost points are selected and vice versa. Three closest points to the 'starting point' are considered, and a line is drawn between a point and the corresponding higher point. Distances between other points on this straight line are calculated, and points are selected if they lie within the margin, γ. If there are no new points within the margin, the higher point is rejected as an outlier. A new point is selected, which has the maximum count above the threshold. This point is considered from the same cluster, and the process is repeated until there are no new points. The maximum length cluster is selected as the lane line, and other clusters are rejected as outliers. This process is visualized in Figure 2-15. Here S is the starting point and point A is rejected as there are no more points in the margin γ. Point B is selected as there are two points within the margin.

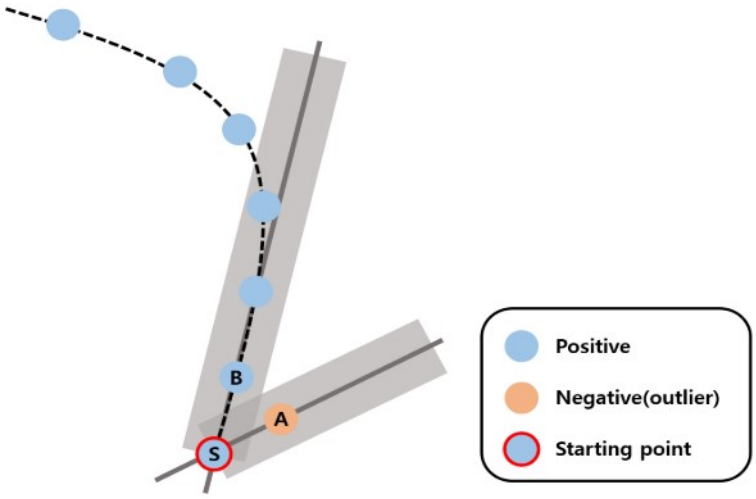


Figure 2-15: Post-processing method suggested by Ko et al. [23]

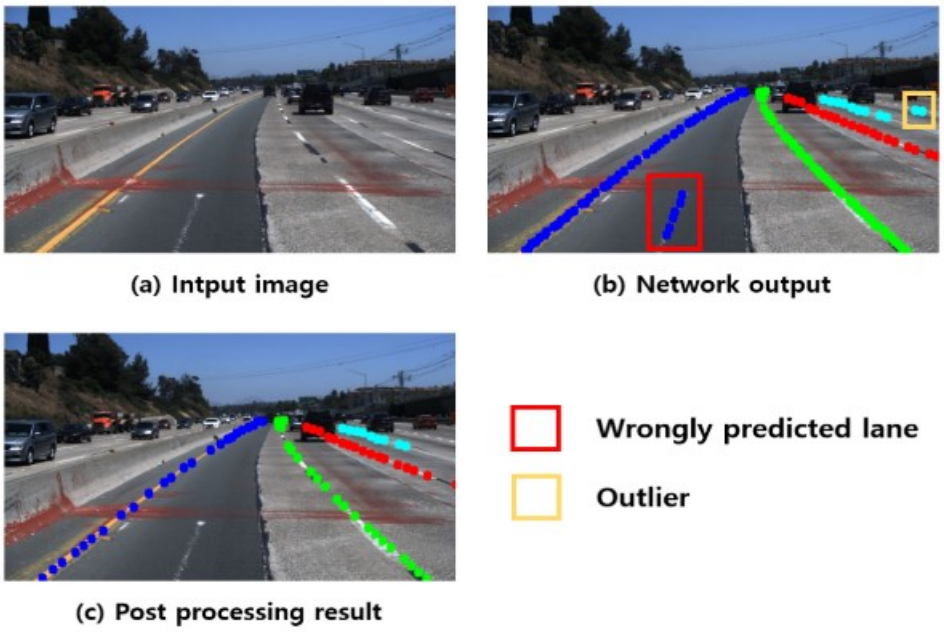


Figure 2-16: Post-processing visualization suggested by Ko et al. [23]

There are two outlier lane lines shown by the red and yellow bounding box in Figure 2-16. These two outliers are rejected as their length is not consistent with other lane lines in the image and ends abruptly.

2-3 CNN-RNN based methods

Recurrent neural network methods include extracting features from an image sequence rather than a single image through convolution neural networks and exploring the temporal relations between the features. Long Short Term Memory (LSTM) unit, convolutional LSTM, Gated Recurrent Unit (GRU), and convolutional GRU are few variants of recurrent network units. A detailed explanation about recurrent neural networks is provided in section 3-2.

2-3-1 Network Architecture

There are two types of recurrent network architectures generally observed in the case of lane detection tasks, where (1) The recurrent module takes the input as a feature map from the convolutional neural network (2) The recurrent module takes the location values of the lane points from the convolutional neural network. Zou et al. propose a feature-map-based method that benefits from both convolutional neural network and the recurrent neural network to solve partial or complete occlusions of the lane line detection [48]. The network utilizes either a U-net or a SegNet style encoder-decoder network which are fully convoluted neural networks [39],[3]. A set of images are taken as input by the encoder network, and the features are extracted. These features are then passed through the Convolutional Long -Short-Term Memory (convLSTM) network, which then propagates the information through different images and produces an output. The equations 2-4 - 2-8 represents the convLSTM operation.

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i) \tag{2-4}$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f) \tag{2-5}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{2-6}$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \odot C_t + b_o) \tag{2-7}$$

$$H_t = o_t \odot tanh(C_t) \tag{2-8}$$

where, $X_1, X_2, ..., X_t$ are inputs, $C_1, C_2..., C_t$ are cell outputs, $H_1, H_2, ..., H_t$ are hidden states, W_{xi}, W_{xf}, W_{xo} are weight matrices and $\odot$ denotes the Hadamard product [17]. $*$ represents convolutional operations [44].

This intermediate output is then utilized by the decoder network, which produces an image of the same size as the input image with the lane detections' probability. ConvLSTMs are used to reduce the computation time and for better performance where the multiplication operations are replaced by convolution operations. Stochastic Gradient Descent (SGD) is used as the optimizer for the network. This network is visualized below in Figure 2-17.

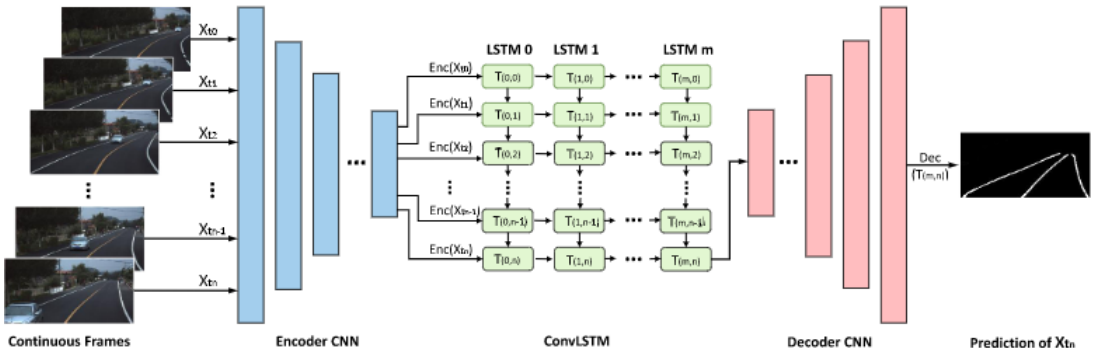


Figure 2-17: Architecture of Robust lane detection network[48]

Yang et al. proposed another lane location value-based detection method that benefits from CNN and RNN [45]. The lane lines are considered as pieces of lanes due to the breaks in the lane strip. It consists of a YOLOv3 (Figure 2-18) network which provides the output vectors containing the center points of a lane piece, the rectangular width and height of the lane piece, and the angle between the two consecutive pieces [37].

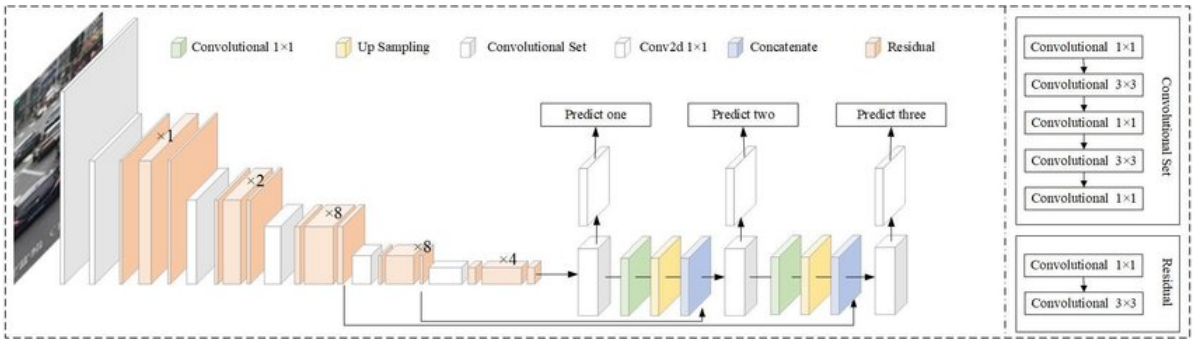


Figure 2-18: Yolov3 network architecture [30]

This output is then passed to the Angle-based Long Short term Memory-Recurrent Convolutional Neural Network (ALSTM-RcNN), a customized LSTM unit consisting of two input gates and two forget gates with one output gate. The ALSTM-RcNN is visualized in Figure 2-19. This also restricts the steering angles in the range $[-15, +15]$ degrees with intervals of 2 to avoid the vehicle's extreme random steering values.

The results from both YOLOv3 and ALSTM-RCNN are fed into a Dempster-Shafer (DS) fusion algorithm to obtain the final values of the output vectors. The DS fusion algorithm is a generalization of the Bayesian theory that combines the outputs from two independent sources by assigning weights to each event's combination. The K-means and Random Sample Consensus (RANSAC) combine the lane pieces and produce a continuous straight line for the detected lanes. This architecture is shown in Figure 2-20.

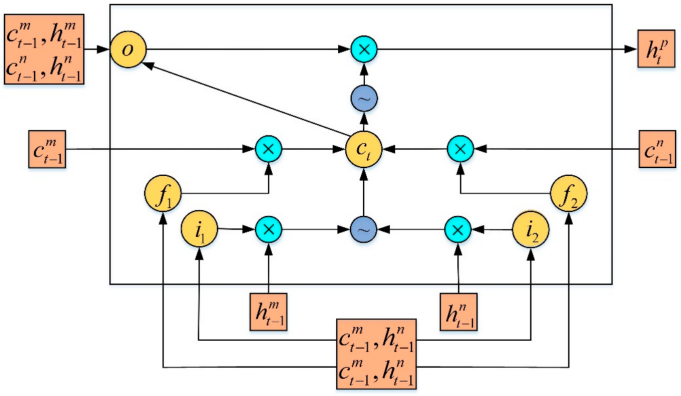


Figure 2-19: Angle based Long Short Term memory (ALSTM) - Recurrent convolutional Neural Network (RCNN) [45]

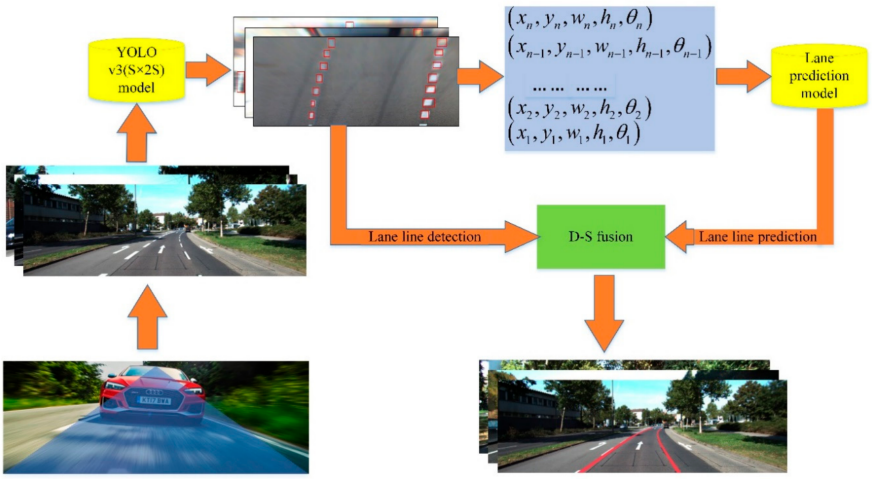


Figure 2-20: Lane Position Detection Based on Long Short term Memory [45]

Another method proposed by Zhang et al. is a feature-map-based method that uses convGRU's as temporal feature extractors [47]. The multiplication operations in GRU 3-2 are replaced with convolutional operations to increase the performance of the convGRUs. The convGRU modules are represented in the Equations 2-9 - 2-12:

$$z^t = \sigma(w^t_z * x^t + U^t_z * h^{t-1} + b_z) \tag{2-9}$$

$$r^t = \sigma(W^t_r * x^t + U^t_r * h^{t-1} + b_r) \tag{2-10}$$

$$\tilde{h^t} = tanh(w^t * x^t + U^t * (r^t \odot h^{\tilde{t-1}}) + b) \tag{2-11}$$

$$h^t = (1 - z^t)h^{t-1} + z^t\tilde{h^t} \tag{2-12}$$

where, $*$ is the convolution operation, z^t is the update gate at layer l at time t, h^t is the final result, r^t is the reset gate at time t, $\odot$ is the elementwise multiplication, $\tilde{h^t}$ is the current

hidden state representation, $h^{\tilde{t-1}}$ is the previous hidden state representation, W_r^t, W_z^t, W and U_r^t, U_z^t, U represent kernel variables, b_z, b_r, b represent biases, x^t is the input feature vector, $\sigma(*)$ is the sigmoid function and $\tanh(*)$ represents the hyperbolic tangent non-linearities. The network includes encoder and decoder networks. A convGRU block is used after the second layer of convolution blocks to extract low-level features such as edges, boundaries, and color. The encoder network's output feature map is passed through multiple convGRU's, which uses spatio-temporal information. The output from multiple convGRU's is then passed through the decoder network, which upsamples the feature map to produce the full-sized image which contains the lane predictions. There are skip connections introduced between the equal-sized encoder and decoder layers. The network is visualized in Figure 2-21. ConvGRU's are simple and robust; they can be trained faster compared to the LSTMs.

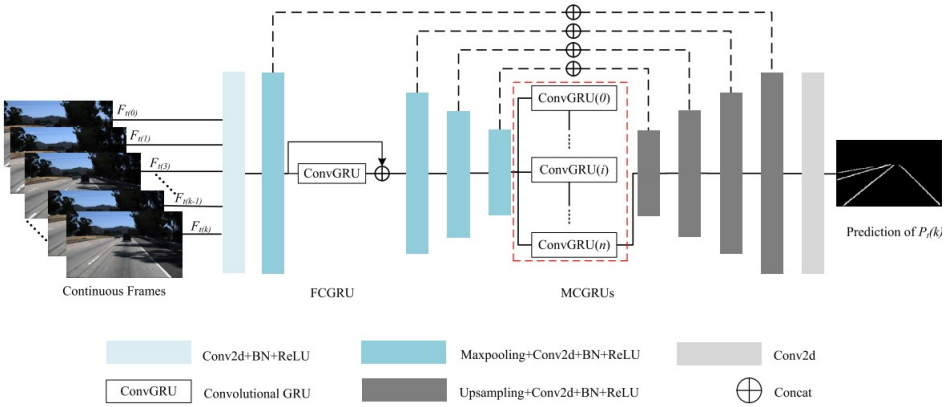


Figure 2-21: ConvGRU network architecture proposed by [47]

Chapter 3

Lane Detection Methodology

For automated/autonomous driving systems in vehicles, robust detection of lanes during challenging scenarios is essential to keep the vehicle centered on the road. The current methods are not completely capable of detecting lanes in challenging scenarios. Relevant information from a sequence of images has to be considered instead of a single image as mentioned in section 2-2 and focus on important features such as lane lines, vehicles over time. Moreover, the automated/autonomous driving system should avoid loosing control over the vehicle due to false positive lane detection.

A model which can process spatio-temporal data from an image sequence intuitively combined with a method which avoids false positive detection is discussed in this chapter. Building blocks of the networks are introduced in sections 3-1, 3-2 and 3-6-1. Choices of baseline networks are explained in section 3-3. A novel spatio-temporal attention method is discussed in section 3-4. A method to reduce false positive lane detection is discussed in section 3-5.

3-1 Introduction to Convolutional Neural Networks

Convolutional Neural Network (CNN) is a special type of network for processing data with a grid-like topology and uses three architectural ideas: local receptive fields, shared weights, and spatial sub-sampling[24]. CNNs produce the output through a series of convolutions on a matrix(e.g., image) with a filter or a kernel. The equation representing the convolutional operation can be written as:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{3-1}$$

Where w is a weighting function, t is the time index, w is the kernel, and a is the age of measurement, and x are the inputs which can be multidimensional arrays of data [14]. Kernels are a multidimensional array of parameters that are usually trained by backpropagation. The

local receptive fields can learn important features from the images, such as corners, shapes, and high-level features. The kernel parameters are shared with all input values in a given single layer. Hence, the CNN has shared weights. This reduces the storage requirements of the system. The sparse connectivity and shared weights increase the convolutions' efficiency for detecting edges in an image. CNNs can process images with large dimensions, and some CNNs can process variable image sizes. A Convolutional Neural Network (CNN) is a powerful tool that can be used to solve computer vision tasks.

3-2 Introduction to Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a type of deep learning methods which process sequential data. They can scale to longer sequences of variable size. The input $x^{(t)}$ of an RNN can be image data, speech data, or text data, and the output $o^{(t)}$ can be updated as shown in Equations 3-2 - 3-5 below:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{3-2}$$

$$h^{(t)} = \tanh(a^{(t)}) \tag{3-3}$$

$$o^{(t)} = c + Vh^{(t)} \tag{3-4}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}) \tag{3-5}$$

where U,V,W are weight matrices, b and c are bias vectors, $y(t)$ is the normalized probability of the output and h is the hidden state at time t.

RNN's have a special property that it can be realized as different input and output types such as:

1. Recurrent Neural Network having connections from one input to one output used for image classification.
2. Recurrent Neural Network having connections from many inputs to one output used in action recognition tasks.
3. Recurrent Neural Network having connections from one input to many outputs used in the tasks such as text sequence generation.
4. Recurrent Neural Network having connections from many inputs to many outputs used to generate subtitles for an image.

These different types of inputs and outputs for RNNs are visualized in Figure 3-1.

One problem with the Recurrent Neural Networks is that there is a possibility of exploding or vanishing gradients during backpropagation due to the error terms' multiplication while updating the weights. If the initial weights are set as zero, and during the backpropagation,

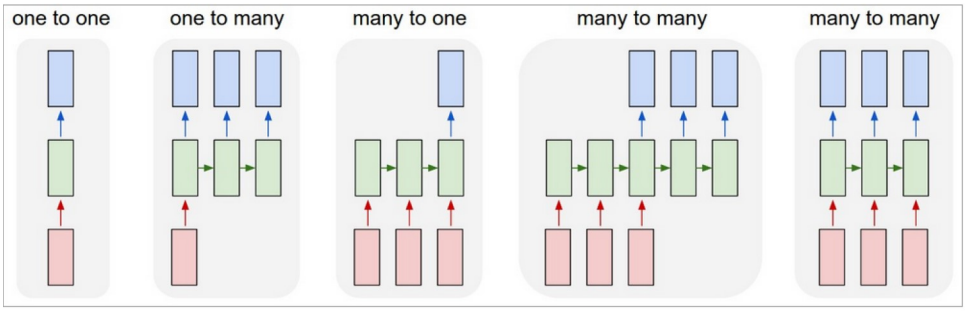


Figure 3-1: Types of Recurrent Neural Networks [2]

the consecutive error terms keep getting smaller over time and vanish after certain iterations. The network cannot be trained further during the vanishing gradient condition. Hence, neural networks such as Long Short Term Memory (LSTM) units or Gated Recurrent Units (GRU) can be used to avoid this.

LSTM

The Long Short Term Memory unit is a recurrent unit with more parameters and gates to control the information flow. It has an input gate, forget gate, and an output gate. It also has a hidden state, a memory state that can be updated over the network's propagation through time by passing the input to the network. Equations 3-6 - 3-11 refer to one step of LSTM.

$$f^{(t)} = \sigma(b^f + U^f x^{(t)} + W^f h^{(t-1)}) \tag{3-6}$$

$$i^{(t)} = \sigma(b^i + U^i x^{(t)} + W^i h^{(t-1)}) \tag{3-7}$$

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^t \odot \tilde{c^{(t)}} \tag{3-8}$$

$$\tilde{c^{(t)}} = g(W^c x^{(t)} + U^c h^{(t-1)} + b^c) \tag{3-9}$$

$$h^{(t)} = o^{(t)} \odot g(c^{(t)}) \tag{3-10}$$

$$o^{(t)} = \sigma(b^o + U^o x^{(t)} + W^o h^{(t-1)}) \tag{3-11}$$

where $x^{(t)}$ is the current input, $h^{(t)}$ is the current hidden vector, g is typically hyperbolic tangent function, $\odot$ is the Hadamard multiplication b^f, U^f, W^f are biases, input weights and recurrent weights for the forget gates. b^i, U^i, W^i are biases, input weights and recurrent weights for the input gate, b^c, U^c, W^c are biases, input weights and recurrent weights for the current value of the memory state $C^{(t)}$. $\tilde{C^{(t)}}$ is the internal memory state, $o^{(t)}$ is the output gate, $s^{(t)}$ is the state unit. b^o, U^o, W^o are biases, input weights and recurrent weights for the output gate [13]. An LSTM unit is visualized in Figure 3-2. Here, the input (IN) is summed with a memory cell (C) and new-memory cell ($\tilde{C}$). The output of $\tilde{C}$ is summed with a memory cell via forget gate (f). The output (OUT) is obtained from C via activation function and output gate (o).

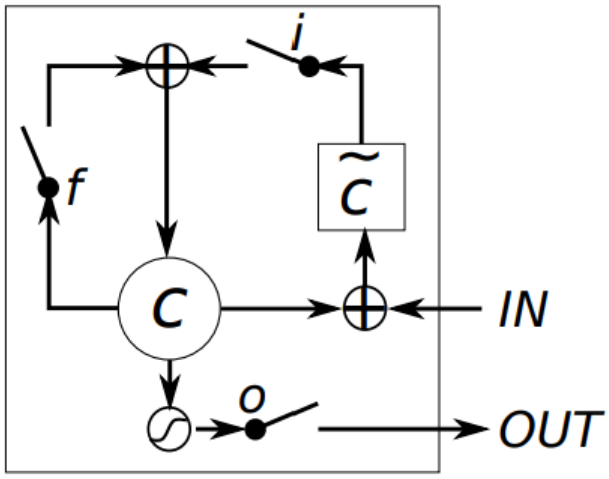


Figure 3-2: Long Short Term Memory unit [10]

GRU

The Gated Recurrent Unit (GRU) differs from the control gates of the unit when compared to the Long Short Term unit. The GRU contains two gates, namely:

1. Reset gate: which decides the amount of past information that is relevant to the current unit.
2. Update unit: decides the amount of past information that has to be passed on to the succeeding units.

$$h_t = (1 - z^{(t)}) \odot h^{(t-1)} + z^{(t)} \odot h^{\tilde{(t)}} \tag{3-12}$$

$$h^{\tilde{(t)}} = g(W^h x^{(t)} + U^h(r^{(t)} \odot h^{(t-1)}) + b^h) \tag{3-13}$$

$$z^{(t)} = \sigma(W^z x^{(t)} + U^z h^{(t-1)} + b^z) \tag{3-14}$$

$$r^{(t)} = \sigma(W^r x^{(t)} + U^r h^{(t-1)} + b^r) \tag{3-15}$$

Equations 3-12 - 3-15 represent the operation of GRUs, where $z^{(t)}$ is the update gate, $r^{(t)}$ is the reset gate, $h^{(t)}$ and $h^{\tilde{(t)}}$ are internal memory state and current memory state. Gated Recurrent Unit is visualized in Figure 3-3. Here, the input (IN) is passed to candidate activation ($\tilde{h}$). The output of $\tilde{h}$ is passed to hidden activation (h) via update gate (z) and then passed back to $\tilde{h}$ via reset gate (r). The output of h is connected to itself via z. The output (OUT) is obtained from h.

A single image input to the neural network is not sufficient if there is dazzle lighting, partial or complete occlusion. For example, consider two vehicles ahead of the ego vehicle and perform the overtaking operations; during the operation, the lane lines are not visible due to complete occlusion. During this, the lanes will be impossible to detect, and the vehicle might

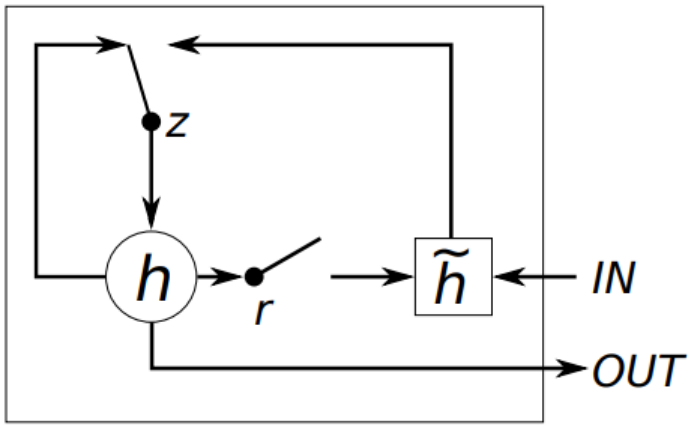


Figure 3-3: Gated Recurrent Unit [10]

lose control. Over-exposure of the camera to the sun's light can also cause the lane lines to be invisible. Hence, there is a need to include temporal information from more than a single image in the neural network to infer the lane lines.

3-3 Baseline Neural Network

The baseline neural network is a skeletal network used to implement a simple and straightforward deep learning based method to extract the lanes. Towards this, two methods are considered initially. The lane pixel classification task and lane location prediction task. Theoretically, any number of lanes can be detected via classification. However, when predicting the lane locations as a set of values from the neural network, the number of values that can be predicted is fixed. Considering the adaptivity to different road conditions and the generalizability of the network, a classification network is considered. In this network, the image is downsampled for feature extraction and upsampled for pixel-wise classification.

The aim of the baseline network is to build a network with all basic entities needed for lane detection. The network should be capable of accepting RGB images and outputting an image containing lane lines. UNet network is considered as the baseline neural network. It is a small fully convolutional network with skip connections and requires less samples [39]. The UNet architecture is visualized in Figure 3-4

UNet architecture contains an encoder-decoder structure. The encoder contains a series of downsizing blocks. The downsizing blocks contain convolution and max-pooling layers which downsize the images from the input image of size $[256 \times 128]$. A downsizing block is visualized in Figure 3-5. Information in the image is encoded in each downsizing layer and meaningful information such as road, vehicles, and environment are extracted in the first layer. Subsequently, the structural elements such as lines, complex shapes are extracted in the next layers [39].

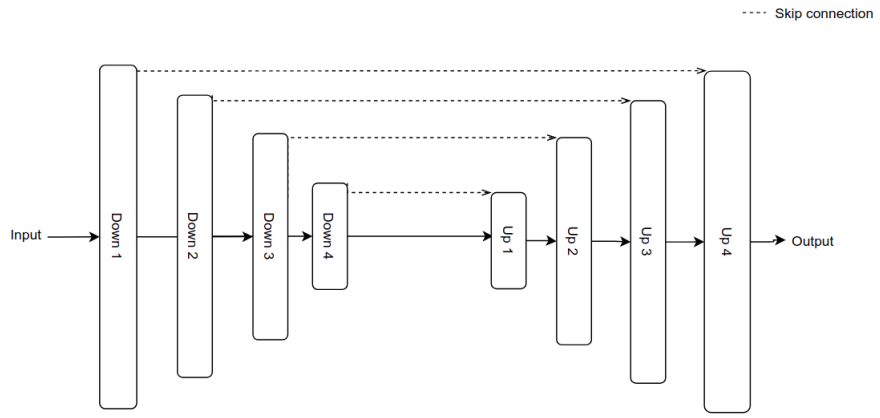


Figure 3-4: UNet Architecture [39]

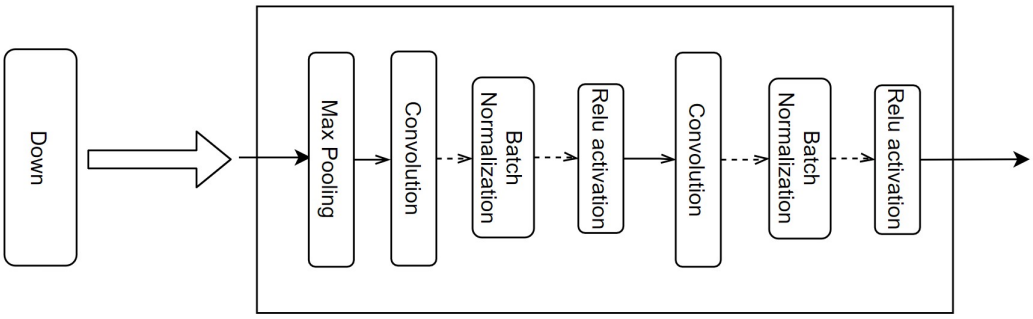


Figure 3-5: Downsizing block of UNet architecture

The output from the last downsized layer is then passed to the decoder which upsizes the image and infers relevant information to detect lane lines. There are four upsizing layers which contain a series of bilinear upsampling layers and convolution layers. The final layer outputs the lane lines with the original image size. Skip connections are used after each downsizing layer and before the corresponding upsizing layer. The upsizing layer is visualized in Figure 3-6

3-4 Spatio - Temporal Attention Model

Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) based temporal methods use hidden units, input gates, forget gate, and update gate to capture information over time. However, due to the fixed memory of hidden units, these methods are capable of learning from a fixed length of image sequence[4]. Also, LSTM/GRU methods do not explicitly place importance on one-(part) of the input over the others. Consider the example frames collected under the tvtLane dataset as shown in Figure 3-7.

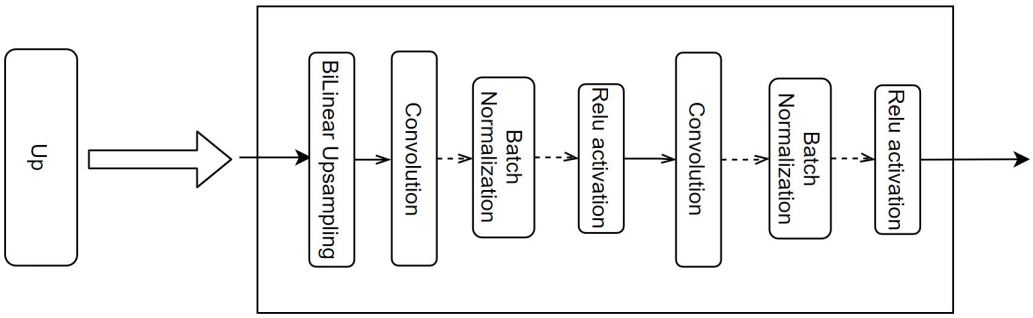


Figure 3-6: Upsizing block of UNet architecture

Figure 3-7: Example frames collected under tvtLane dataset as a part of TuSimple dataset[48]

An occlusion is caused by the overbridge over the lane lines in the last three frames (Figure3-7). The first two frames from the left provide most of the relevant information to detect lane lines. LSTM/GRU methods having three hidden units may not be able to produce lane detection in the last three frames. Even when a LSTM module having five hidden units is used. The results may not be as accurate as just using the first two frames. This is because the LSTM/GRU methods do not use explicit weights to decide the importance of the input images. Hence, there is a need for a method to learn longer temporal information dependecies between input frames and place importance over relevant images.

Cognitive attention is the ability to focus on one thing and ignore other things. A neural network can be used to mimic cognitive attention. A novel attention method is proposed in which an attention network learns to focus on important parts of the input by assigning weights to each input and particular parts of the input. In the example mentioned previously, the attention method can extract important features over a long temporal distance and focus on relevant features without any limit. Attention model uses the hidden output produced by a temporal feature extractor such as LSTM/GRU's.

Hidden outputs of a previous time are multiplied with the input of the current time step. Since the LSTM/GRU's tend to better represent the input, the hidden unit can focus around the features of the current input. The hidden output of the previous time step and the input at the current time step can be combined using a set of weights. An activation of these weights can then be obtained to learn which image-(part) is important. A sum of the product of weights and input images can be obtained to extract the important features. These features can be imported to a temporal feature extractor to obtain the output at the current time step and the hidden output. Equations 3-16 - 3-21 denote the equations for the attention

mechanism explained above.

The attention model is applied when the input image sequences are downsized and the features are extracted by a series of convolution layers. Consider an input vector of features, Output of downsized convolution $Down_4$, Figure 3-3, Input, $x_t = \{x_1, x_2, x_3, ..., x_n\}$, Hidden vectors, h = $\{h_1, h_2, h_3, ..., h_n\}$, then:

$$x_t = (x_{down4(t)} * k_{in})(i) \tag{3-16}$$

$$z_{t-1} = Ux_t + Vh_{t-1} \tag{3-17}$$

$$w_t = Pr(Wz_{t-1}|x_t) \tag{3-18}$$

$$\bar{x} = \sum_{t=1}^{n} w_t x_t \tag{3-19}$$

$$x_o, h_{t-1} = F(\bar{x}, h_{t-2}) \tag{3-20}$$

$$x_{out} = (x_o * k_{out})(i) \tag{3-21}$$

Here, $(*)$ is the convolution operation over index i. k_{in} and k_{out} are the kernels of size 1×1 with 1 and 512 channels, respectively. F can either be LSTM/GRU. The learnable weights U, V, W can be a variable of size 1×1 or a vector of size of input (8×16) which can be represented as 1×128 or a fully connected layer of size 1×128.

3-4-1 Temporal Attention Model

In the temporal attention model, the learnable weights U, V, W in Equations 3-16 - 3-21 are constants. These constants place importance over a particular input. The importance of each image is measured by a probability value of constant. Figure 3-8 represents the temporal attention model.

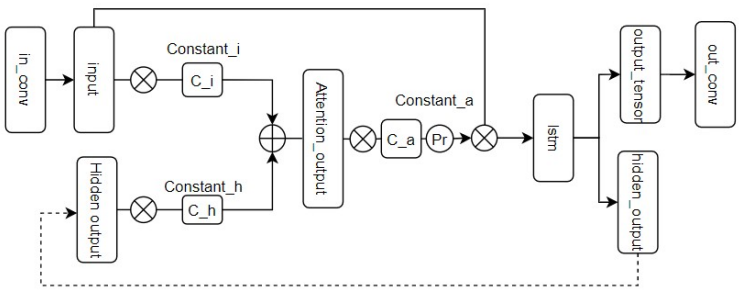


Figure 3-8: Temporal attention model

3-4-2 Spatio Temporal Attention Model

The spatio - temporal attention model has three learnable weight matrices of size 8×16. Each weight matrix is multiplied with the input feature matrix, the previous hidden output, and the attention output of the current step. The attention model learns the importance of the individual pixel with the weights and hidden layer without the knowledge of neighbouring pixels. Figure 3-9 represents the spatio temporal attention model.

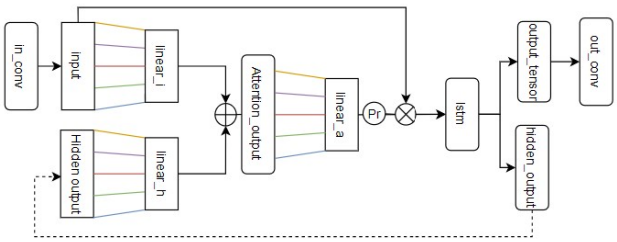


Figure 3-9: Spatio temporal attention model

3-4-3 Spatio Temporal Attention Model with Fully Connected Layers

The spatio temporal attention model with fully connected layers is similar to the spatio temporal attention model. However, each learnable weight is multiplied with all values of the input feature matrix as shown in Figure 3-10 below. The idea here is to extract spatial dependencies between the pixels in the same image while extracting temporal features. Figure 3-10 represents the spatio temporal attention model with fully connected layers.

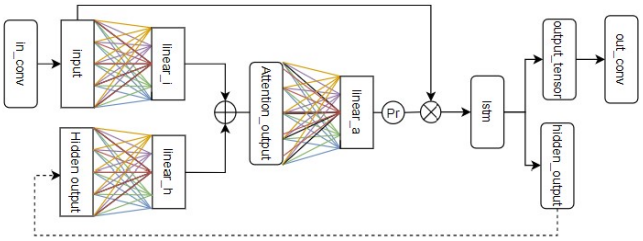


Figure 3-10: Spatio temporal fully connected attention model

Detailed network architecture parameters are shown in Table 3-1. The attention model is implemented after the fourth down-layer and before the first up-layer. The attention model is modular in nature and can be used with other network architectures such as SegNet and Fully convolutional network [27].

Table 3-1: Detailed architecture parameters of UNet with attention layer
*Layer 1-1, 1-2, 1-3: Learnable Constant or Learnable array or fully connected layer

Layer		Input	Output	Kernel	Stride	Pad	Activation
Inconv	Conv 1-1	3X128X256	64X128X256	3x3	1	1	ReLU
	Conv 1-2	64X128X256	64X128X256	3x3	1	1	ReLU
	MaxPool	64X128X256	64X64x128	3x3	2		
Down 1	Conv 2-1	64x64x128	64X64x128	3x3	1	1	ReLU
	Conv 2-2	64X64x128	128X64x128	3x3	1	1	ReLU
	MaxPool	128X64x128	128X32x64	3x3	2		
Down 2	Conv 3-1	128X32x64	128X32x64	3x3	1	1	ReLU
	Conv 3-2	128X32x64	256X32x64	3x3	1	1	ReLU
	MaxPool	256X32x64	256X16x32	3x3	2		
Down 3	Conv 4-1	256X16x32	256X16x32	3x3	1	1	ReLU
	Conv 4-2	256X16x32	512X16x32	3x3	1	1	ReLU
	MaxPool	512X16x32	512X8x16	3x3	2		
Down 4	Conv 5-1	512X8x16	512X8x16	3x3	1	1	ReLU
	Conv 5-2	512X8x16	512X8x16	3x3	1	1	ReLU
Attention	Conv 6-1	512X8x16	1x8x16	1x1	1		
	*Layer 1-1	1x128	1x128				
	*Layer 1-2	1x128	1x128				
	*Layer 1-3	1x128	1x128				
	LSTM	128	128				
	Conv 6-2	1x8x16	512x8x16	1x1	1		
Up 1	Bilinear Upsampling	512x8x16	512X16x32	2x2	2		
	Conv 7-1	1024X16x32	256X16x32	3x3	1	1	ReLU
	Conv 7-2	256X16x32	256X16x32	3x3	1	1	ReLU
Up 2	Bilinear Upsampling	256X16x32	256X32x64	2x2	2		
	Conv 8-1	512X32x64	128X32x64	3x3	1	1	ReLU
	Conv 8-2	128X32x64	128X32x64	3x3	1	1	ReLU
Up 3	Bilinear Upsampling	128X32x64	128X64x128	2x2	2		
	Conv 9-1	256X64x128	64X64x128	3x3	1	1	ReLU
	Conv 9-2	64X64x128	64X64x128	3x3	1	1	ReLU
Up 4	Bilinear Upsampling	64X64x128	64X128x256	2x2	2		
	Conv 10-1	128X128x256	64X128x256	3x3	1	1	ReLU
	Conv 10-2	64X128x256	64X128x256	3x3	1	1	ReLU

3-5 Reduction of False Postives

The number of false positives are an important metric in the case of lane detection. False positives can essentially cause an abrupt change in the lane. This can cause discomfort to the passengers of the vehicle or the failure of the semi-/autonomous vehicle systems [29]. One of the main causes for the increase in the false positives, lack of features or information in the network which causes the network to make poor generalization and prediction.

The lack of information can be attributed to the maxpooling layer. It is used in the network to downsize the feature maps without learning the operation during the training process and is computationally less expensive. However, it stores only the max value inside the moving kernel during operation. Hence, introducing convolutional layers to downsize the image can be an effective strategy to avoid the loss of information. The convolution layer can also learn the operation which can help in extracting more meaningful information [41]. The convolution operation is a sliding kernel operation which is similar to maxpooling layer. An example of convolution operation in comparison to the maxpooling layer is shown in Figure 3-11. Here, smaller colored windows represent the kernels of the layers.

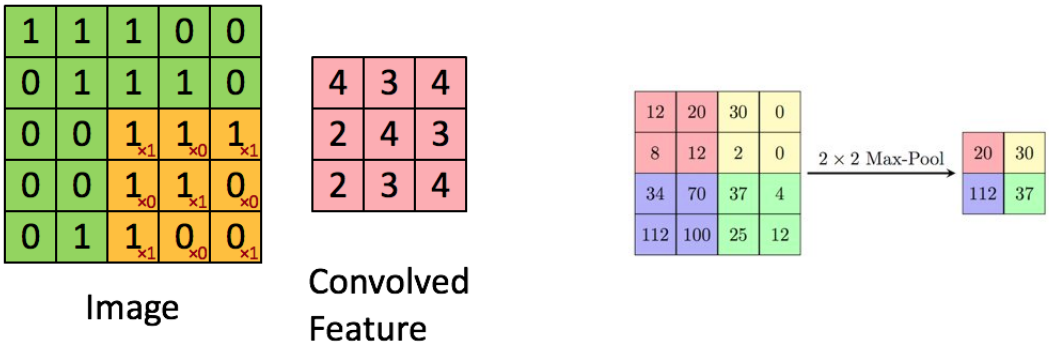


Figure 3-11: Example convolution and maxpooling operations

The maxpooling layer in Figure 3-11 (right) stores the values 20, 30, 112, and 37 out of the other values in the matrix. This can cause the network to generalize poorly. On the other hand, the convolution operation embeds all values in the matrix Figure 3-11 (left). It is also easier to replace the maxpooling layer as the operations are similar. The output sizes of the maxpooling layer and convolution layer given the input size can be calculated as:

$$output\_size = \frac{input\_size - kernel\_size}{stride} + 1 \tag{3-22}$$

$$output\_size = \frac{input\_size - kernel\_size + 2 * (padding)}{stride} + 1 \tag{3-23}$$

Here, Equation 3-22 represent the output size of the max pooling layer for a given kernel size and stride. Equation 3-23 represent the output size of the convolution operation for a given kernel size, padding, and stride values. When the padding is zero, the output size of the convolution operation will be equal to the output size of the maxpooling operation for the same kernel size and stride. Four maxpooling layers are implemented in the network, one after each downsampling layer 3-1. Replacing the convolutional operation after the first downsampling layer can be more effective as shown by Kaiming et al. [15].

3-6 Training setup

To train the above-mentioned networks, loss function and optimizers are required. A brief summary and choice of various loss functions, optimizers, and other necessary parameters are explained further.

3-6-1 Loss Function

A loss function is required to adjust the trainable parameters in the network. Probabilistic loss and regression losses are two main types of loss functions. Probabilistic losses provide a probability value of whether a particular pixel in an image belongs to a given label class. Different types of probabilistic losses include binary cross-entropy loss, categorical cross-entropy loss, sparse categorical cross-entropy loss, and Kullback-Leibler (KL) divergence loss. Regression losses predict a numerical value which is an approximation of the true value. Regression loss examples include mean absolute error, mean squared error, cosine similarity, mean squared logarithmic error, and many others.

In the lane detection task, the lane image pixels can be classified as either 'lane' or 'not lane'. Since there are only two classes, binary cross-entropy loss is considered [16]. Binary cross-entropy loss is a specialized form of cross-entropy loss which is formulated in the Equation 3-24.

$$L_{bce} = -\frac{1}{M}\sum_{m=1}^{M}[y_m * log(h_\theta(x_m)) + (1 - y_m) * log(1 - h_\theta(x_m))] \tag{3-24}$$

where M is number of training examples, y_m is target label for training examples m, x_m is the input for training example m, h_θ is the model with neural network weights θ. A weighted binary cross-entropy is used to balance the loss function during training. The weight of 0.02 is chosen for lane class and 1.02 is chosen for not-lane.

3-6-2 Optimizer

To train the network, the weights (θ) of the network needs to be optimized. Loss function L_{bce} defined in the previous subsection 3-6-1 is used to optimize the network. Gradient Descent is a classical method to optimize the weights (parameters). It can be defined as:

$$\theta_i := \theta_i + \Delta\theta_i \tag{3-25}$$

where

$$\Delta\theta_i = -\frac{\alpha\partial J(\theta)}{\partial\theta} \tag{3-26}$$

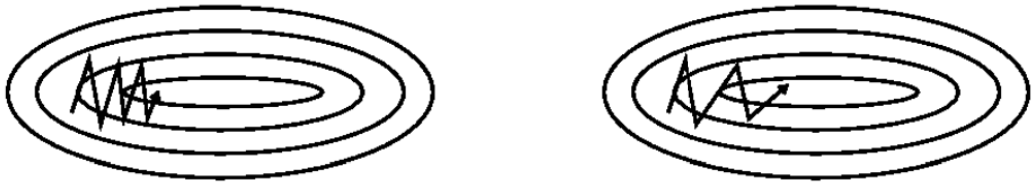
Where $J(\theta)$ is the Loss function L_{bce}, θ are the weights of the network, $\Delta\theta_i$ is the gradient at a given step. The most important aspect of gradient descent is that it is continuous and

differentiable [40].

To optimize millions (13.39 million for UNet baseline model) parameters of the network, all images from a large dataset are considered in the gradient descent algorithm. This is especially time consuming and redundant as each update requires going through all data repetitively. Hence, Stochastic Gradient Descent (SGD) is considered [35]. SGD overcomes the redundancy by performing gradient calculations on one sample (image). SGD is defined as:

$$\theta := \theta - \eta \nabla_\theta J(\theta) \tag{3-27}$$

SGD fluctuates with high variance and is susceptible to converging at local minima as visualized in figure 3-12 (a) and 3-13 respectively.



(a) SGD without momentum (b) SGD with momentum

Figure 3-12: Fluctuation of SGD without/with momentum[40]

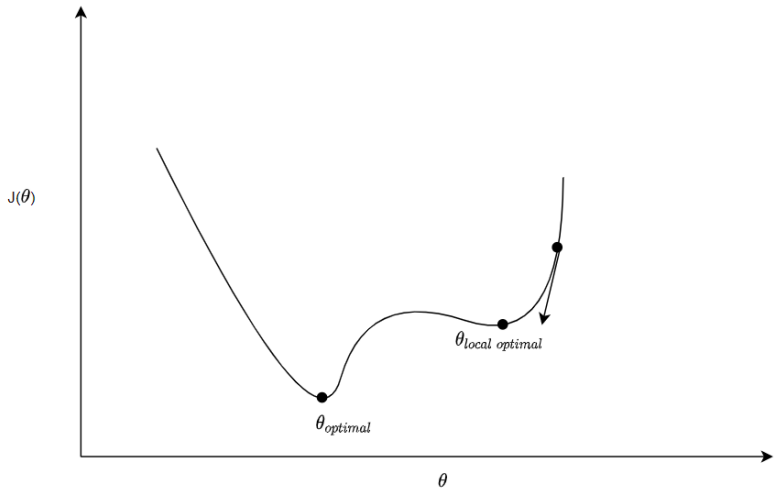


Figure 3-13: Gradient descent over the cost function $J(\theta)$

Hence, a momentum term is added to overcome this. Momentum helps accelerate SGD in the correct direction and dampens the fluctuation. A fraction (γ) of past gradients is added to the current update vector. Hence, the parameters are updated as follows.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \tag{3-28}$$

$$\theta_i := \theta_i - v_t \tag{3-29}$$

Here v_t refers to the momentum term and is usually set to 0.9. The momentum term increases for weights whose gradients point in the same direction and reduces updates for weights whose gradients change direction.

Adaptive Moment Estimation (ADAM) stores the exponentially decaying average of past squared gradients. v_t and exponentially decaying average of past gradients m_t. The terms are defined as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3-30}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3-31}$$

where m_t is the first moment estimate or mean, v_t is the second moment estimate or variance. When m_t and v_t are initialized as zero vectors, they are biased towards zero. To correct these, a bias correction term is used.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3-32}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{3-33}$$

Hence, the parameter update can be yielded as:

$$\theta_{t+1} = \theta - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon} \hat{m}_t \tag{3-34}$$

for β_1 and β_2 values of 0.9 and 0.999 are proposed by Kingma et al. [22]. Although, SGD is known to converge to global minima, it can sometimes converge to local minima. An experiment is conducted to measure the difference between ADAM and SGD on the proposed UNet spatio-temporal based attention network. The network is trained on the tvtLane dataset. The graph 3-14 shows that SGD tends to settle at local minima and hence does not reach an optimal minimum. The SGD based method reaches 97.5% and ADAM based method reaches 97.80% validation accuracy under the same settings, and the validation accuracy curve of ADAM is much smoother in comparison with SGD, which denotes stability. The ADAM optimizer is used to train subsequent models.

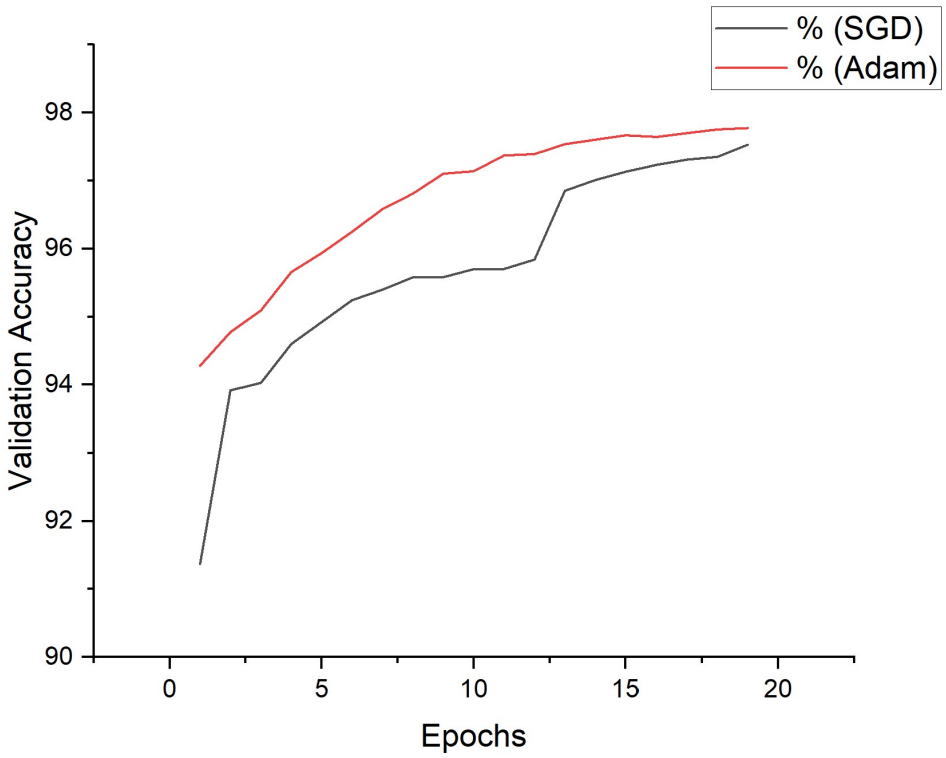


Figure 3-14: Validation accuracy of UNet spatio-temporal attention based network trained on tvtLane dataset using Adam/SGD

Chapter 4

Experiments and Results

In this chapter, the lane detection datasets that are used in the training of the proposed networks are explained. Experimentation details of various methodologies (Chapter 3) are discussed in detail and the results are presented.

4-1 Lane Detection Datasets

Large-scale open-source datasets are often necessary to train deep learning networks. This is because deep neural networks have a high number of training parameters. There are many datasets available for lane detection tasks, and more datasets are being added. Some of the large-scale datasets such as TuSimple, LLAMAS, and tvtLane datasets are discussed below, and the performance of state-of-the-art networks on these datasets.

4-1-1 TuSimple Dataset

TuSimple dataset contains 7000 one-second long video clips of 20 frames each. The training set contains 3626 video clips with 3626 annotated frames. The view direction of the camera is close to the driving direction. Polylines are used for lane markings. The maximum number of lane markings in any given frame is 5. However, in most frames there are 4 lane markings. The extra lane marking is used during the lane changing operations. The lanes are around the center of sight, which is essential for autonomous cars. Accuracy is used as the main method to evaluate the deep learning method [12]. The location of the recorded images is not available. An example of the TuSimple dataset is shown in the Figure 4-1.

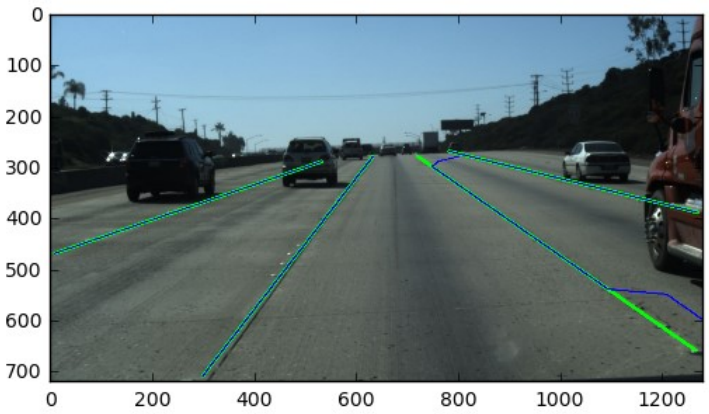


Figure 4-1: Example of the TuSimple dataset along with the lane markings [12]

4-1-2 tvtDataset

tvtlane dataset combines TuSimple dataset images and rural roads' images of China [48]. A set of 20 consecutive images are stored as a sequence. The resolution of images is 128×256. The data distribution of the training and test set is shown in the Table 4-1. The images in the training set are flipped and rotated in three degrees to obtain the augmented dataset and are labelled as "f" and "3d" respectively, in the dataset. A total of 38.192 images are created from 19.096 recorded images.

Table 4-1: Distribution of training and test dataset in tvtDataset [48]

	TuSimple(highway) labelled images	Rural road images	Total
Train dataset	7252	2296	9548
Test dataset	540	728	1268

Table 4-2 shows the sampling of images in the sequence of tvtLane dataset with strides 1, 2, and 3. Different strides of images represent the varying speed of the vehicle.

Table 4-2: Image samples in tvtLane training set [48]

Stride	**Sampled frames**	**Labelled Image**
1	9, 10, 11, 12, 13	13
2	5, 7, 9, 11, 13	13
3	1, 4, 7 ,11, 13	13
1	16, 17, 18, 19, 20	20
2	12, 14, 16, 18, 20	20
3	8, 11, 14, 17, 20	20

Two test sets are created, the first test set is used to test the overall performance of the network, and the second test set is created to test the robustness of the network. In the first test set, there are 540 sequences and every 13^{th} and 20^{th} image is labelled. The second test set consists of 12 types of hard scenes which are labelled.

The authors suggest the usage of accuracy and F1-measure as the evaluation metrics for the dataset. An example of the tvtlane dataset image and the corresponding ground truth is visualized in Figures 4-2 and 4-3.

Figure 4-2: Example of the tvtlane dataset image [48]

Figure 4-3: Example of the tvtlane dataset ground truth [48]

4-1-3 LLAMAS dataset

The LLAMAS dataset is an unsupervised lane marker dataset consisting of 100.042 images with labeled lane markers. The dataset is collected by Bosch, United States [6]. Human effort is used only in collecting lidar and camera data and to remove faulty images. Both lidar and camera data are collected and optimized to generate the lane detection labels. The recording mainly consists of 14 highways and around 25 km in each of the highways. The resolution of the greyscale images is 1280×717. The dataset is divided into 58.269 training images, 20.844 validation images, and 20.929 testing images. The labels contain 2D, 3D dashed lines which are pixel level lane associations. Behrendt et al. suggest average precision (AP) for binary segmentation on a per-pixel basis [6]. An example of the LLAMAS dataset is observed in Figure 4-4.

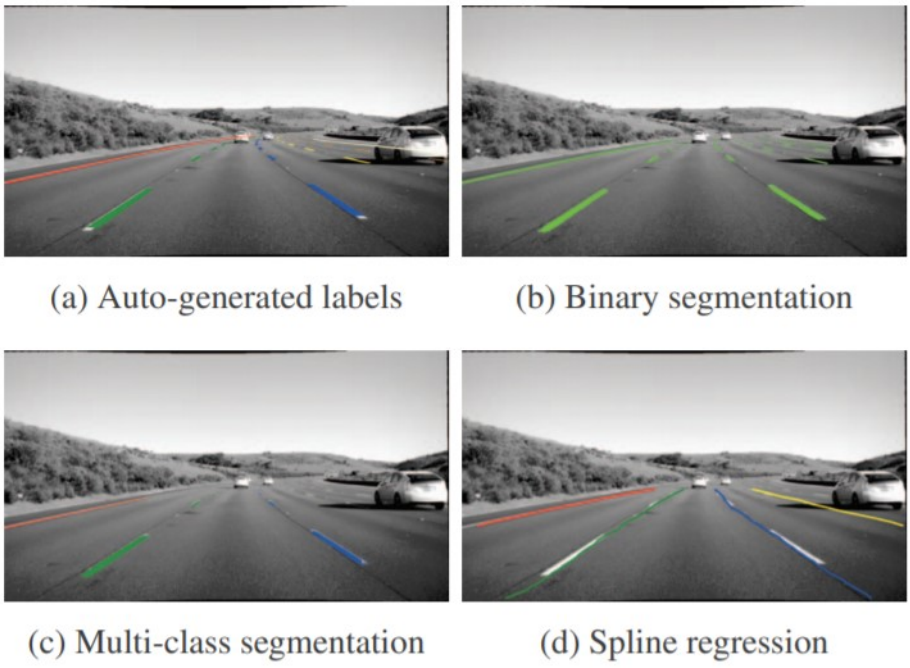


Figure 4-4: Example of the Llamas dataset[6]

Table 4-3 show the dataset training samples of LLAMAS dataset used in training. The training dataset consists of both the sampled frames and the total number of training samples in LLAMAS dataset is 101,149 image sequences.

Table 4-3: Image samples in LLAMAS dataset

Stride	Sampled frames	Labelled Image	No. of train samples
1	1, 2, 3, 4, 5	5	46000
1	2,4,7,11,16	16	58149

Table 4-4 shows the number of images in each dataset, the resolution of each image, and the road type in the three datasets under consideration. The images in higher resolution are downsized to 128×256 before training the network.

Table 4-4: Summary of different lane detection datasets

Dataset	Number of images	Resolution	Road-type
TuSimple	13.200	1280x720	Highway
LLAMAS	100.042	1276x717	Highway
tvtLane	38.192	128x256	Highway and rural

4-2 Evaluation Metrics

To assess the network capability in performing the lane detection task on an unseen dataset, evaluation metrics are useful. Lane detection tasks are evaluated mainly through metrics such

as accuracy, F1-measure, and Average Precision. These metrics are discussed below in detail.

4-2-1 Accuracy

The performance metric for TuSimple and tvtLane dataset is accuracy. Accuracy is defined as the ratio of true predictions among the total number of outputs.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4-1}$$

where, TP = True positive, TN = True negative, FP = False postive, FN = False negative

4-2-2 F1-measure

The performance metric for the tvtLane dataset is F1-measure. Precision and recall are the building blocks of F1-measure. The union of the correctly predicted lane pixels, given that the labels are true positives. The union of incorrectly predicted lane pixels, given that labels are counted as false positives.

A certain threshold is set for these unions, above which the predictions are viewed as the predictive value of a particular class. The general F-measure is given as:

$$F - measure = (1 + \beta^2)\frac{Precision * Recall}{\beta^2 * (Precision + Recall)} \tag{4-2}$$

where precision and recall are given as:

$$precision = \frac{TP}{TP + FP} \tag{4-3}$$

$$recall = \frac{TP}{TP + FN} \tag{4-4}$$

Where TP = True positives of lane class, TN = True negatives of non-lane class, FP = False positives of lane class, FN = False negatives of non-lane class. When $\beta = 1$ in Equation 4-2, it is known as F1-measure which is the harmonic mean of precision and recall values. Equation 4-5 represents F1-measure.

$$F1 - measure = \frac{Precision * Recall}{Precision + Recall} \tag{4-5}$$

4-2-3 Average Precision (AP)

The performances for LLAMAS dataset are evaluated by average precision. Average precision (AP) is the average of precision over recall values between 0 and 1 [9]. Equation 4-6 represent the average precision.

$$AP = \sum_{p=1}^{U}(\sum_{q=1}^{V+1}(Precision_p * \Delta Recall_q)) \tag{4-6}$$

where $\Delta Recall$ is the difference between two consecutive values of recall. $Recall_0$ is set to 0, $Precision_0$ is set to 1 and variable V is set to 100 [47].

4-2-4 General setting of Training

The neural network trainings are carried out on LISA computing service provided by SURF. LISA computing service has LINUX based machines with NVIDIA GTX $2080Ti$ GPU clusters with high volume and high speed scratch drive. The network is trained using only one $2080Ti$ gpu as the other combinations did not yield better accuracy during training and testing.

Stochastic Gradient Descent (SGD) and Adaptive Moment estimation (ADAM) optimizers are considered during the training purposes. To converge to the global minimal loss solution for the complex mapping function of the neural network, a step-based learning rate is used. After each epoch, the learning rate is reduced, $lr_{next} = lr_{prev} * gamma$, where $gamma \in \{0, 1\}$. Having a constant learning rate causes the complex mapping function of the neural network to reach a suboptimal local minimum loss. Plot 4-5 shows the decrease in learning rate over each epoch.

The parameter setting for the experiments is shown in Table 4-5

Table 4-5: Training paramters for neural networks

Training Parameters	**Value**
Stochstic Gradient Descent (SGD)	momentum=0.9, learning rate=0.01
Adaptive Moment (ADAM)	learning_rate=0.01
Learning rate step (StepLR)	Step_size=1 epoch, gamma=0.9
Binary weighted cross-entropy	class weights = {0.02 (not-lane), 1.02 (lane)}
Batch size	16

4-3 Experiments

In this section, the network details and experimental methods of baseline network, UNet temporal attention network, UNet spatio-temporal attention network, and UNet spatio-temporal attention network with fully connected layer. Moreover, the experiments are conducted using tvtLane dataset as it is a large dataset with various challenging scenarios compared to TuSimple and LLAMAS dataset. The networks performing well on the tvtLane dataset are

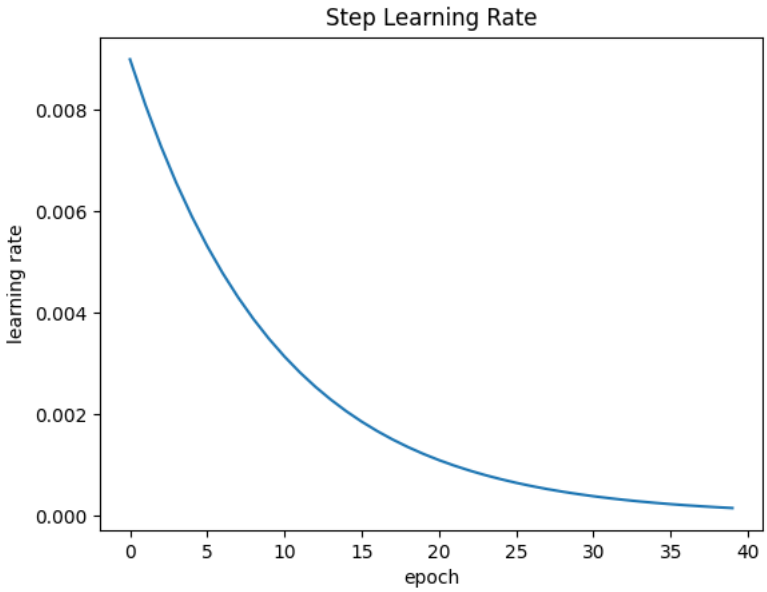


Figure 4-5: Plot illustrating learning rate decrease over epochs

also experimented on the TuSimple dataset and the LLAMAS dataset. The results of these experiments are elaborated in section 4-4

4-3-1 Experiment 1

In this experiment, the baseline architecture, which is an encoder-decoder based UNet architecture as discussed in section 3-3 is conducted. This network is used as a benchmark to compare with the variants of UNet attention-based methods. This network is trained for 40 epochs on the tvtLane dataset, beyond which the validation accuracy converged to a single value. Accuracy, precision, recall, and F1-measure are used as metrics to evaluate the network. The network is visualized in Figure 4-6.

4-3-2 Experiment 2

In this experiment, temporal-based attention model is implemented. The attention model is placed after the fourth downlayer. The layers 'Down 1' to 'Down 4' process the images in sequence and are input to the attention model. The input in the attention model is multiplied with a constant value C_i and the hidden input from the previous time step is multiplied with a constant value C_h, these values are combined and mutilpied with a constatn value C_a to obtain the attention weights. The output feature map from the attention model is then upsampled through layers 'Up 1' to 'Up 4'. The network is visualized in Figure 4-7.

The network is trained for 35 epochs until the validation accuracies do not change in value. Figure 4-8 shows the validation accuracy against the number of epochs (number of single passes through the entire dataset) of the network during training.

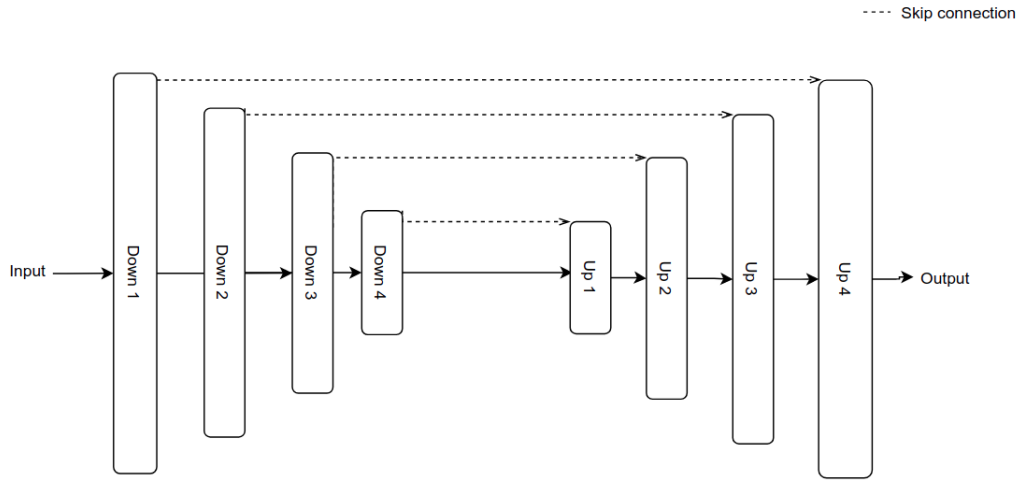


Figure 4-6: Baseline UNet network

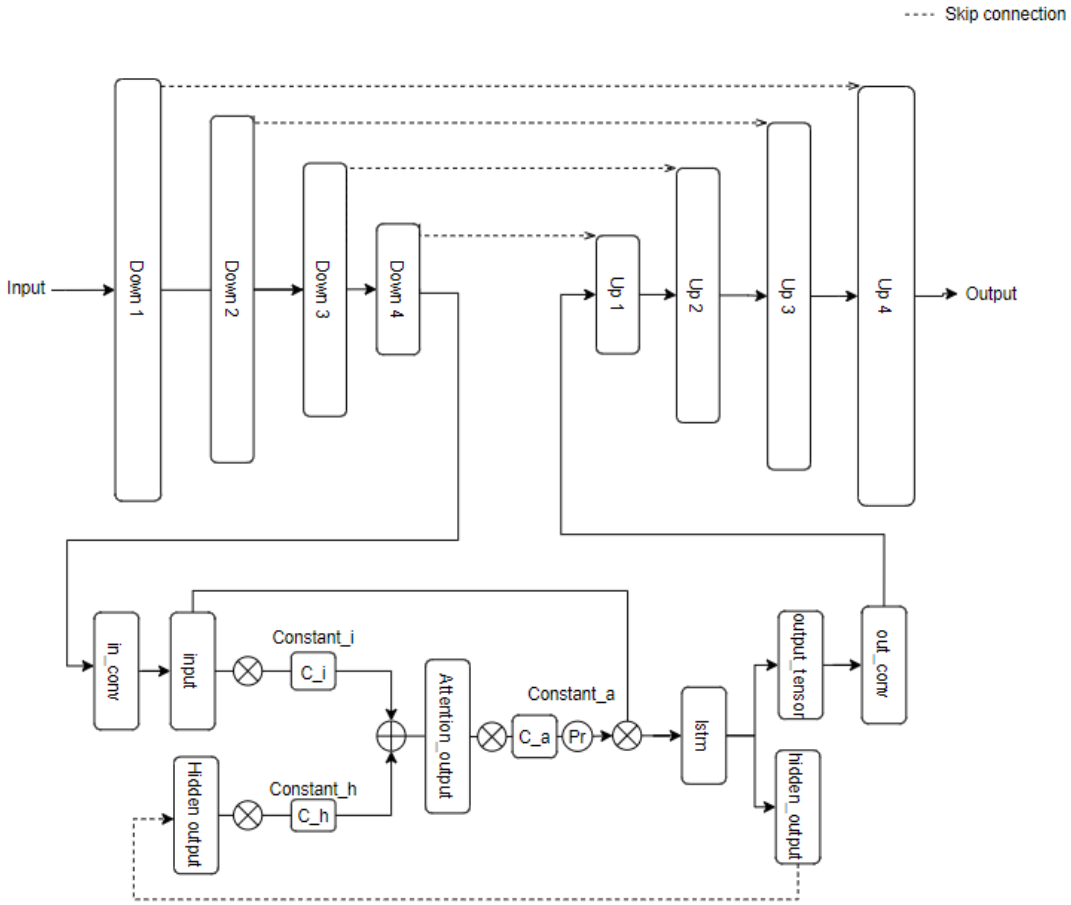


Figure 4-7: UNet temporal attention network

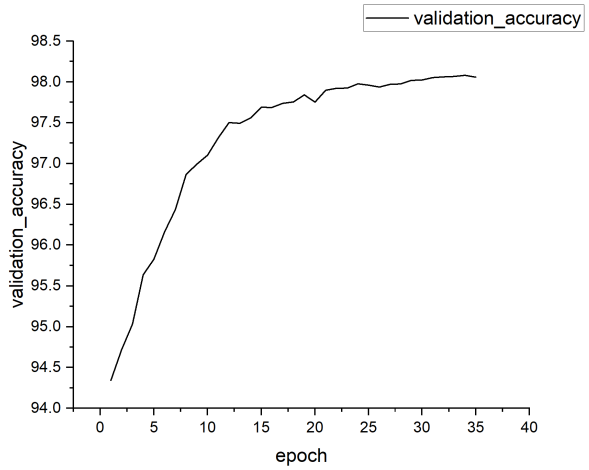


Figure 4-8: Validation accuracy during training of UNet temporal attention network

4-3-3 Experiment 3

In this experiment, a spatio-temporal-based attention model is implemented. The attention model is placed between the fourth downsampling layer and the first upsampling layer. The layers 'Down 1' to 'Down 4' process the images in sequence and are input to the attention model. The input, the hidden input from the previous time step, and the combined values of input and hidden values in the attention model are multiplied with a matrix of size 8×16 to obtain the attention weights. The output feature map from the attention model is then upsampled through layers 'Up 1' to 'Up 4'. The network is visualized in Figure 4-9.

The network is trained for 35 epochs until the validation accuracy does not change in value. The Figure 4-10 shows the validation accuracy against the number of epochs (number of single passes through the entire dataset) of the network during training.

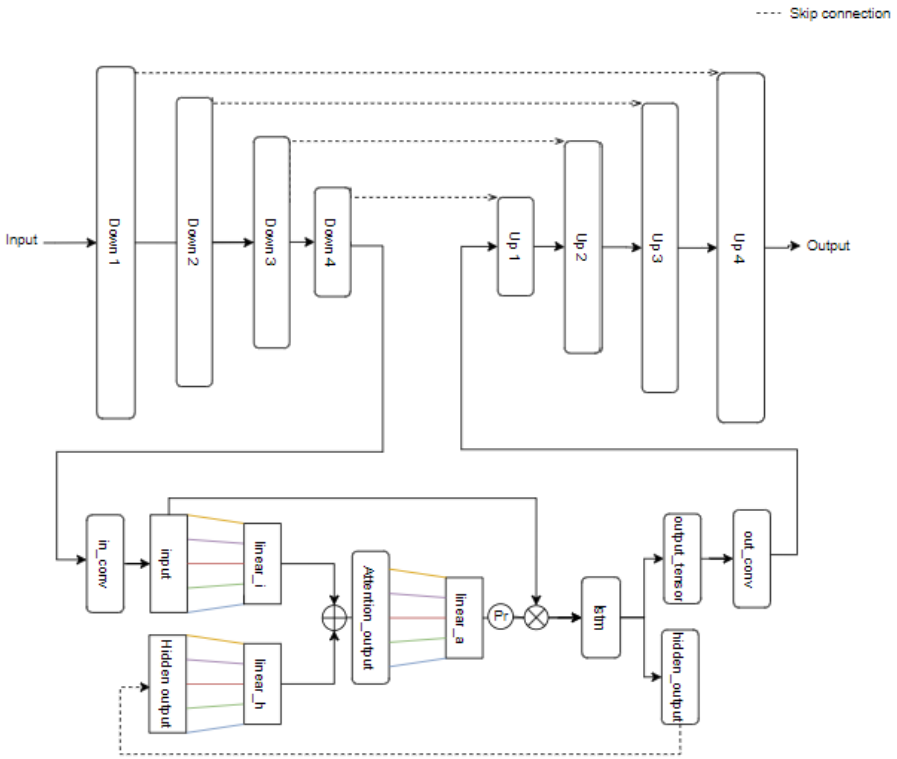


Figure 4-9: UNet spatio-temporal attention network

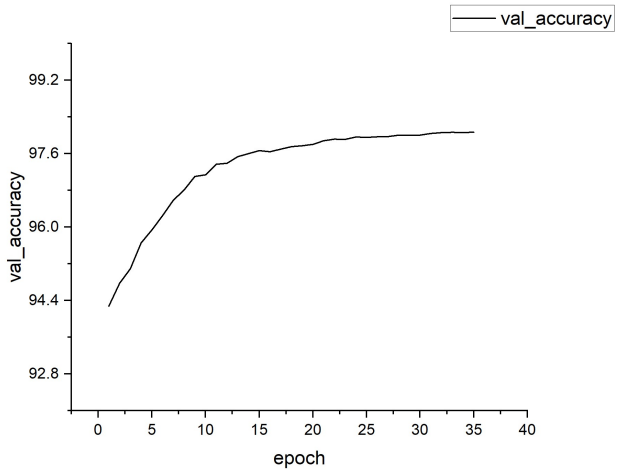


Figure 4-10: Validation accuracy during training of UNet spatio-temporal network

4-3-4 Experiment 4

In this experiment, UNet spatio temporal fully connected attention network is implemented. The attention model is placed after the fourth downlayer. The layers 'Down 1' to 'Down 4' process the images in sequence and are input to the attention model. The input, hidden input from the previous time step and the combined outputs of input and hidden values in

the attention model are multiplied with a fully connected vector of size 1×128 to obtain the attention weights. The output feature map from the attention model is then upsampled through layers 'Up 1' to 'Up 4'. The network is visualized in Figure 4-11.

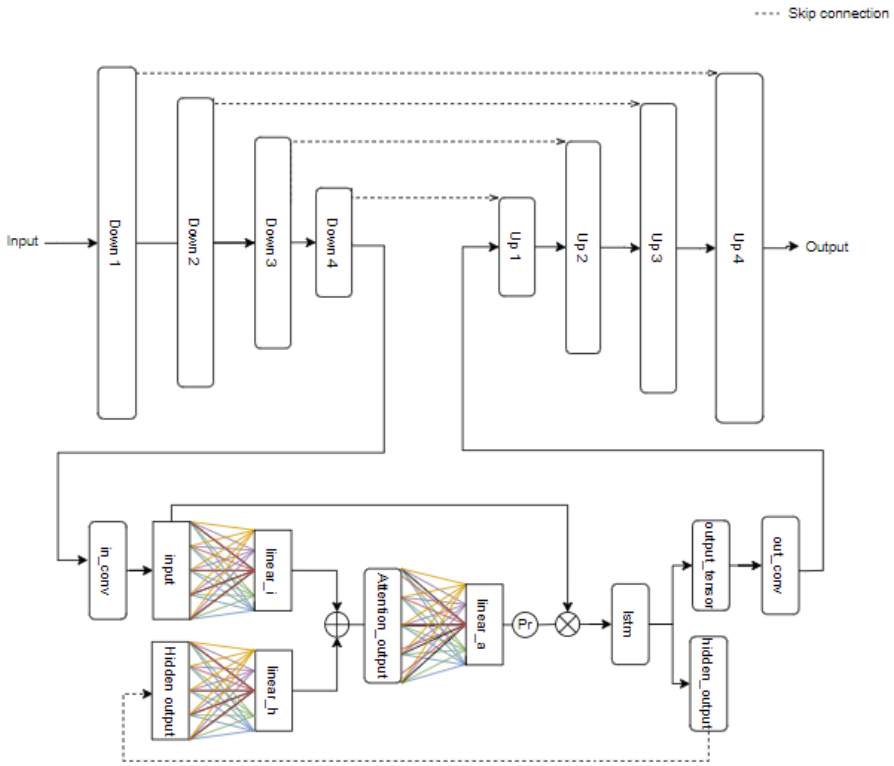


Figure 4-11: UNet spatio-temporal attention network with Fully connected layers

The network is trained for 35 epochs until the validation accuracies do not change in value. The Figure 4-12 shows the validation accuracy against the number of epochs (number of single passes through the entire dataset) of the network during training.

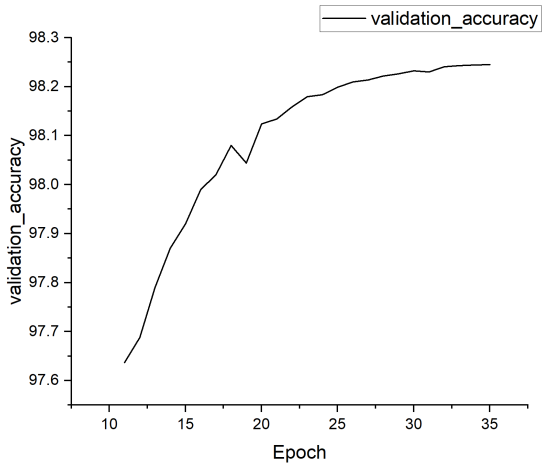


Figure 4-12: Validation accuracy of UNet spatio temporal FC attention network during training

4-3-5 Experiment 5

In this experiment, to observe the effect of maxpooling layer in reducing the false positives, the maxpooling layer in the first downsampling layer is replaced with the convolution layer with a kernel size of 2 and a stride of 2. UNet spatio temporal FC attention network is trained on the tvtLane dataset for 35 epochs. Figure 4-13 shows the replacement of the maxpooling layer in the first downsampling layer of the network. UNet spatio temporal FC attention network is trained with the replacement. The validation accuracy during training is observed in the Figure 4-14.

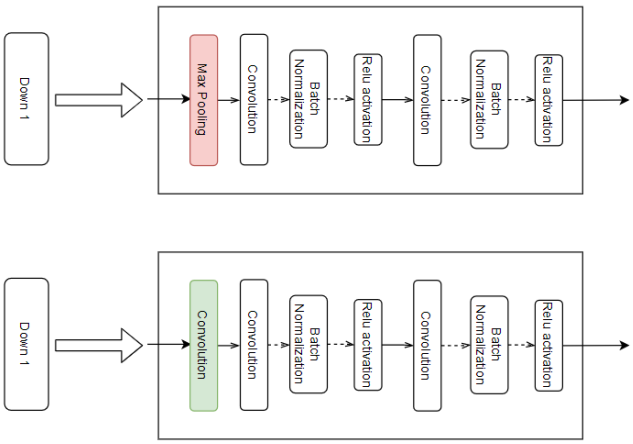


Figure 4-13: Maxpooling layer replaced by the convolution layer at the first downsampling layer

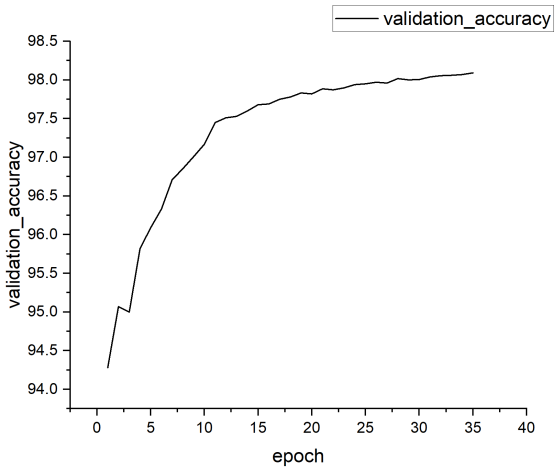


Figure 4-14: Validation accuracy of UNet spatio temporal FC attention network where, max-pooling layer is replaced by convolution layer at the first downsampling layer

4-4 Results

The proposed networks in the previous section 4-3 are trained on tvtLane and LLAMAS datasets independently. These trained networks are evaluated on TuSimple, tvtLane, and LLAMAS datasets. A qualitative analysis on the Netherlands dataset [43] is also carried out in chapter 5 . Augmented tvtLane dataset has a high number of training samples of rural, urban, and highway roads, therefore all proposed networks are primarily trained and evaluated on tvtLane dataset. A few of the best models are then trained/evaluated on other datasets. The evaluation metrics of the proposed networks are compared with state-of-the-art networks and the results are presented below.

4-4-1 Results on tvtLane dataset

Evaluation metrics on the tvtLane test dataset 1 include accuracy, precision, recall, and F1-measure, and the evaluation metrics on tvtLane dataset 2 include precision scores. The evaluation metrics of the proposed network in comparison with the baseline network and other state-of-the-art methods are shown in Table 4-6 and Table 4-7.

In the case of tvtLane test dataset 1, UNet spatio temporal attention network shows the best overall performance with the highest precision score and F1-measure. This network has higher test accuracy compared to state-of-the-art networks and comparable recall value. Precision value is an indicator of false positives. Higher precision scores indicate lower false positives. The proposed network is able to reduce false positives for tvtLane dataset. The UNet spatio temporal attention with the first maxpooling layer replaced with the convolution layer achieves a high recall but does not produce a significant improvement over the other proposed networks in terms of precision. Hence, introducing a convolution layer for down-sampling is not effective in the proposed network. A detailed analysis of tvtLane test datasets 1 and 2 are explained in chapter 5.

Table 4-6: Performance on tvtLane dataset 1
* proposed networks,
**UNet spatio temporal attention FC network with first maxpooling layer replaced with convolutional layer

Network	Accuracy (%)	Precision	Recall	F1-measure
Baseline network	98.0770	0.8807	0.9159	0.8979
SegNet[48]	96.9300	0.7960	0.9620	0.8710
UNet ConvLSTM[48]	97.9428	0.8525	0.9502	0.8987
SegNet ConvLSTM[48]	97.7800	0.8520	0.9640	0.9010
UNet temporal Attention*	98.0800	0.8771	0.9365	0.9058
UNet Attention w/ conv layer**	97.9871	0.8653	**0.9409**	0.9015
UNet spatio temporal Attention*	98.1443	**0.8873**	0.9356	**0.9108**
UNet spatio temporal FC Attention*	**98.1816**	0.8831	0.9359	0.9088

Table 4-7: Precision score of networks tested on tvtLane dataset 2
* proposed UNet based networks

Network	1-curve & occlude	2-shadow	3-bright	4-occlude	5-curve	6-dirty & occlude
Baseline network	0.7018	0.7441	0.6717	0.6517	0.7443	0.3994
SegNet	0.6810	0.7067	0.5987	0.5132	0.7738	0.2431
UNet_ConvLSTM	0.7591	0.8292	0.7971	0.6509	0.8845	0.4513
SegNet_ConvLSTM	0.6545	0.6937	0.5748	0.5405	0.7385	0.2846
temporal Attention*	0.7938	0.8743	**0.8013**	0.7014	**0.8894**	0.5215
spatio temporal Attention*	**0.8430**	**0.8909**	0.7732	0.5740	0.8322	0.4692
spatio temporal FC Attention*	0.8239	0.8782	0.7646	**0.7031**	0.8871	**0.5295**
	7-urban	**8-blur & curve**	**9-blur**	**10-shadow**	**11-tunnel**	**12-dim**
Baseline network	0.4422	0.7612	0.8523	0.7881	0.7009	0.5968
SegNet	0.3195	0.6642	0.7091	0.7499	0.6225	0.6463
UNet_ConvLSTM	**0.5148**	0.8290	**0.9484**	0.9358	0.7926	**0.8402**
SegNet_ConvLSTM	0.2885	0.5774	0.6868	0.7584	0.6978	0.7395
temporal Attention*	0.4935	0.8290	0.8517	0.9286	0.7516	0.8218
spatio temporal Attention*	0.4567	**0.8358**	0.8090	0.9244	0.78931	0.8046
spatio temporal FC Attention*	0.4848	0.7354	0.9023	**0.9395**	**0.8794**	0.7542

In the case of tvtLane dataset 2, there are 12 challenging scenarios which include occlusion, shadow, dirty, rural, and urban roads. UNet spatio temporal network performs better in curve, occluded, shadow, and blurred roads than other methods compared. In most other scenarios, both UNet temporal attention and UNet spatio temporal FC attention networks perform better or similar to state-of-the-art networks. Figure 4-15 shows samples of tvtLane test datasets 1 and 2. UNet spatio temporal FC attention network predicts lanes which are closest to the ground truth and UNet spatio temporal network predicts an extra correct lane in most cases.

Figure 4-15: Sample results on tvtLane testdataset 1 and 2. (a) Input images (b) ground truths (c) UNet-ConvLSTM [48] (d) UNet spatio temporal attention network (e) UNet temporal attention network (f) Unet spatio temporal FC attention network

4-4-2 Results on TuSimple dataset

Evaluation of all proposed networks on metrics such as accuracy, precision, recall, and F1-measure using TuSimple dataset is carried out. Table 4-8 compares the evaluation metrics of all proposed networks with state-of-the-art networks along with the number of parameters in each of the networks. The UNet spatio temporal FC attention network performs better than all other networks in comparison of accuracy and precision. It also attains comparable results in recall and F1-measure. Number of parameters (in millions) in Table 4-8, is an indication of the computation requirement of the network. Lower number of parameters indicates that the network requires lesser computation time and resources. All proposed networks have a significantly less number of parameters while performing better than other networks. The predicted lanes by all proposed networks in comparison with UNet-ConvLSTM by Zou et al. on TuSimple dataset are observed in Figure 4-16 [48]. Although only four lanes are marked in the rightmost ground truth frame in Figure 4-16, the UNet spatio temporal FC attention network is able to detect upto 6 lane lines in the frame.

Table 4-8: Performance comparison of proposed networks with state-of-the-art networks on TuSimple Dataset
*proposed networks
**average value calculated by two values from test accuracy [48]

Network	Accuracy	Precision	Recall	F1-measure	Params (M)
PINET(64x32)[23]	97.6200	0.8470	0.8950	0.8710	17.9
Res18-Qin[36]	96.9000	0.6300	0.6910	0.6590	58.4
Res34-Qin[36]	96.8900	0.6370	0.7010	0.6680	98.9
SCNN[34]	96.7900	0.6540	0.8080	0.7220	76.9
LaneNet	97.9400	0.8750	0.9270	0.9010	78.8
SegNet[32]	96.0450	0.7300	0.9810	0.8380	117.9
SegNet ConvLSTM **	97.9550	0.8520	0.9640	0.9010	268.9
Baseline network	97.9600	0.8640	0.9550	0.9080	5.0
Unet double ConvGRU[47]	98.0400	0.8735	**0.9525**	**0.9113**	13.4
UNetConvLSTM[48]	97.9323	0.8623	0.9192	0.8898	204.6
UNet temporal Attention*	98.0500	0.8759	0.9233	0.8990	13.5
UNet spatio temporal Attention*	98.1383	0.8809	0.9245	0.9022	13.5
UNet spatio temporal FC Attention*	**98.2078**	**0.8861**	0.9503	0.8956	13.6

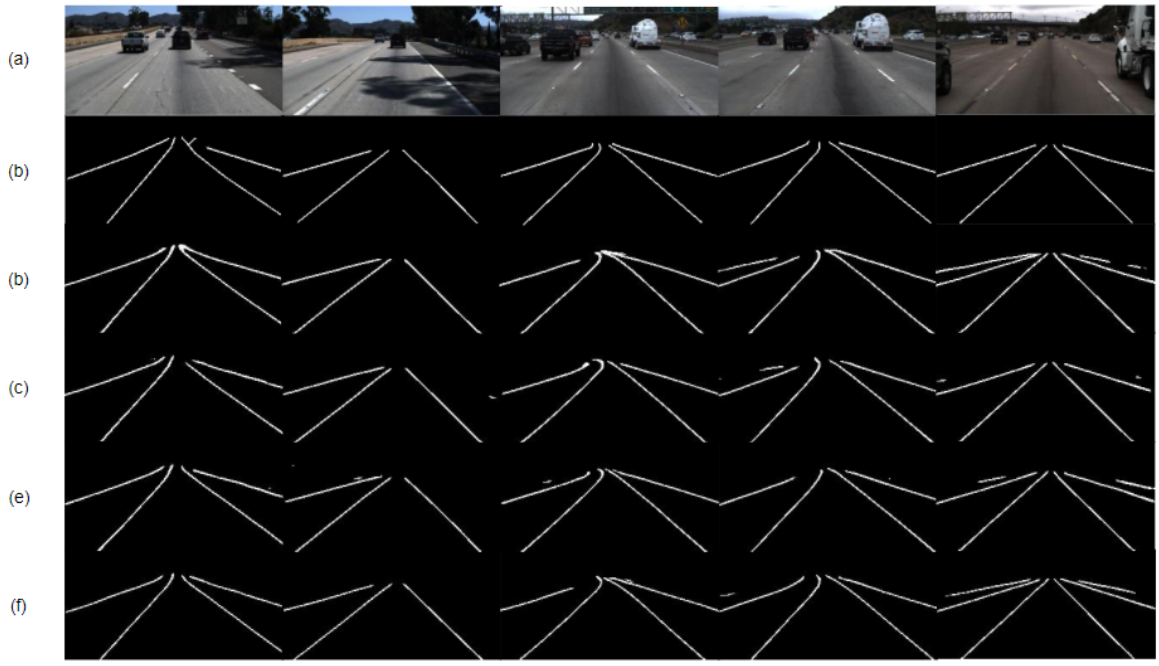


Figure 4-16: TuSimple results on (a)Input images (b) ground truths (c) UNet-ConvLSTM [48] (d)UNet spatio temporal attention network (e) UNet temporal attention network (f)Unet spatio temporal FC attention network

4-4-3 Results on LLAMAS dataset

Evaluation metrics for LLAMAS dataset are Average Precision (AP), corner precision, and corner recall. Overall, UNet spatio temporal network performed better in comparison with

other proposed networks. Hence, it is used to train and evaluate on LLAMAS dataset. Table 4-9 shows the results of UNet spatio temporal attention network in comparison with state-of-the-art networks. The proposed network performs best in corner recall, which is an indication of a lower number of false negatives and has a comparable average precision. However, the lower corner precision indicates that there are a higher number of false positives. A possible explanation for the low corner precision is given in section 5-1.

Table 4-9: Performance comparison of proposed networks with state-of-the-art networks on LLAMAS Dataset *proposed network

Model	Average Precision (AP)	Corner Precision	Corner Recall
UNet double ConvGRU [47]	0.**8519**	**0.6162**	0.6163
SegNet ConvLSTM[48]	0.8500	0.5487	0.6839
UNet ConvLSTM[48]	0.8510	0.5857	0.6558
UNet spatio temporal attenion*	0.8028	0.3246	**0.7183**

The predicted lanes by UNet spatio temporal attention network are visualized in Figure 4-17. In all the top frames, the predicted lane lines are shown in red color and in all bottom frames, only the predicted lane lines are visualized. Visually, there are very few false positives and the lane lines appear to be predicted accurately.

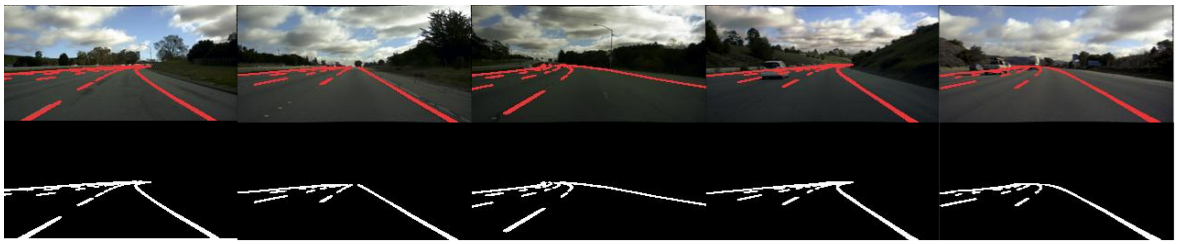


Figure 4-17: LLAMAS detection results on UNet spatio temporal attention network

Chapter 5

Discussion and Conclusion

The thesis aims at improving the lane detection and reducing the number of false positive lane detection. Towards this, the current state-of-the-art methods were studied and gaps in the research were found and are presented in Chapter 1 and 2. To address these research gaps, novel sequence-to-sequence-based methods are proposed in Chapter 3 with the objective to capture relevant information over an image sequence. Experiments were conducted to evaluate the effectiveness of the proposed algorithms in comparison with the existing methods and the results are presented in Chapter 4. These results are explained, the effectiveness of the proposed methods, and the problems faced during the experiments are discussed. A future recommendation is given.

5-1 Discussion

Lane line prediction results between the baseline network and all proposed networks are compared in Figure 5-1. The baseline network (Figure 3-4) is able to produce lane lines with tvtLane dataset 2. However, in the leftmost and the rightmost frame (Figure 5-1), the baseline network is not able to predict the lane lines in the shadow region. However, all proposed networks are able to make predictions in such scenarios. Due to the occlusion in the second frame from left (Figure 5-1), the baseline network is not able to recognize the lane line. In the third and fourth frame from the left (Figure 5-1), due to the less bright environment and blur, the baseline network is not able to make correct predictions.

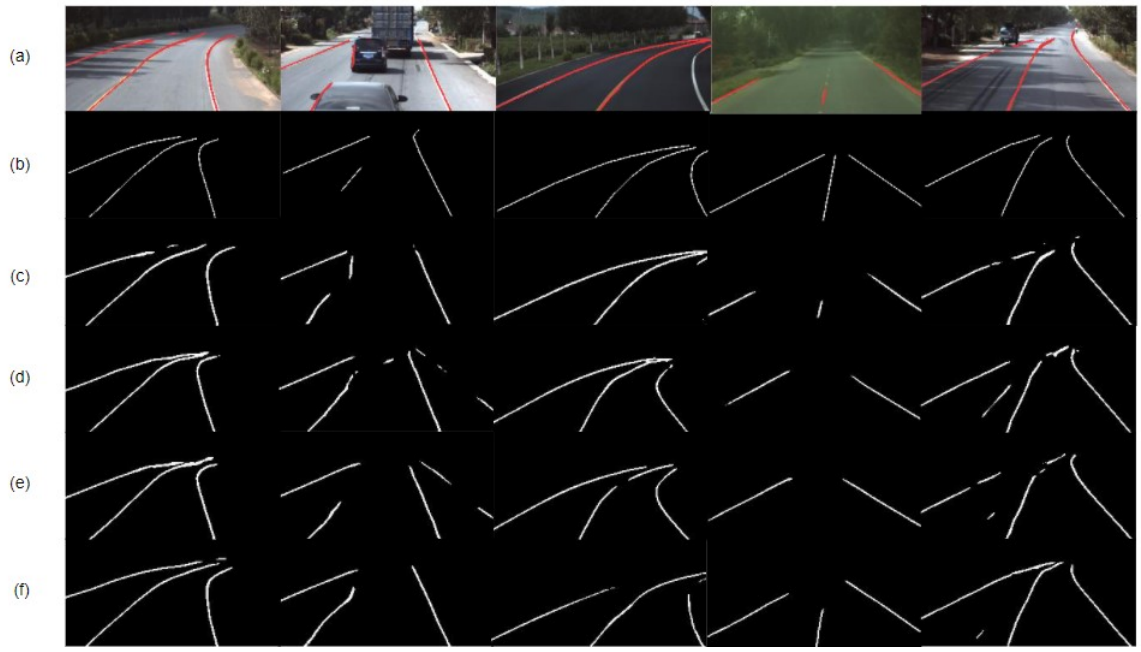


Figure 5-1: (a)Input images (b) Ground truths (c)Predicted lanes for UNet baseline network tested on tvtLane dataset-2 (d) UNet spatio temporal attention network (e) UNet temporal attention network (f) Unet spatio temporal FC attention network

However, all proposed networks take advantage in terms of two main aspects (1) Sequence based image input, (2) Ability to focus on relevant images. When compared to UNet baseline architecture, there is an improvement in the precision metrics on the tvtLane dataset 2 observed in Table 4-7.

The UNet spatio temporal attention network is an upgraded version of UNet temporal attention-based network, where the weights of the attention module are replaced with a learnable matrix of size of the input feature map (Figure 3-8 and 3-9). These weights can focus on both important feature maps (image) and the specific segments of the feature maps. Through this, more meaningful relationships can be learnt between the images in the sequence. In Figure 5-1, UNet spatio temporal attention network is able to produce smoother lane lines compared to UNet temporal attention network, observed in the first and third frames from left.

UNet spatio temporal FC attention network is similar to UNet spatio temporal attention-based network where the attention matrices are replaced by fully connected matrices (Figure 3-9 and 3-10). Here, all input pixels are connected to all output pixels. The fully connected matrices establish the relation between pixels in the given feature map. Hence, the network should be able to learn the intraframe dependencies and the continuous nature of the lane lines. Figure 5-1 shows the predicted lane lines in the images. The lane lines observed here are continuous and there are improvements in the predictions compared to the earlier.

UNet-ConvLSTM network proposed by Zou et al. has a very high test-accuracy of 97.91%, and F1-measure of 0.7114. It is compared with the UNet spatio temporal fully connected attenion network. In Figure 5-2, (a) represents the sequence of lane detection images from

UNet-ConvLSTM. The convLSTM module is not able to focus on the middle lane line on the road in frames 2 & 3. Although frame 1 consists of a partially detected middle lane line, the model is not able to construct a temporal relationship between frames 1-3. Comparatively, Figure 5-2 (b) represents the sequence of lane detection images from the proposed UNet spatio-temporal attention-based model. The attention module is able to focus on the middle lane lines in all frames and produces a favourable result. There is continuity from the end of lane in the first frame to the beginning of the second frame. Hence, the attention-based network is effective in extracting spatio-temporal information.

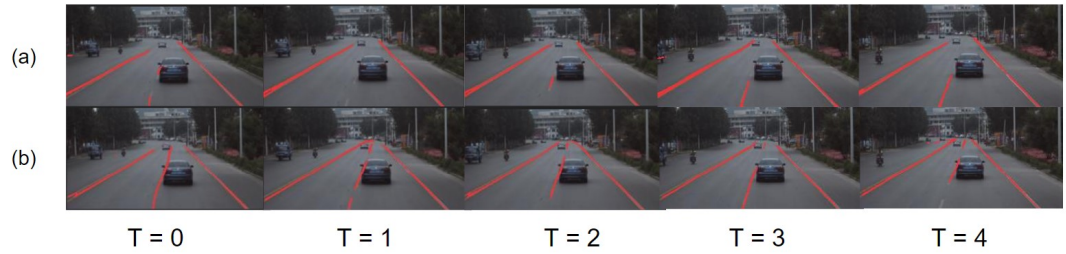


Figure 5-2: Comparison of lane detection from UNet ConvLSTM[48] and proposed UNet Spatio-Temporal FC Attention network

The UNet spatio temporal attention network produces high recall values but low corner precision values on the LLAMAS test dataset. This indicates that the network has low false negatives and high false positives in comparison to state-of-the-art networks (Table 4-9). One reason for the low corner precision values is the preprocessing of LLAMAS dataset for training the proposed network. Figure 5-3 shows the original image and two types of downsized images. The original image with resolution of 1276×717 is downsized to 256×128. During the operation, the lane line values are lowered as a part of the averaging method. This causes the lane lines to be less visible in the left downsized image. During training, the network is unable to learn these lane lines. To overcome this, the lane line values are binarized and scaled to 255, which creates jagged lane lines as shown in the right downsized image. The network is able to learn these lane lines, but does not produce optimal results.

Figure 5-3: Sample of original llamas dataset (top) and downsized images (bottom)

The UNet spatio temporal attention network trained on the LLAMAS dataset is used to test qualitatively on the Netherlands dataset. The Netherlands dataset is a collection of video clips of highways of the Netherlands by Vos et al.[43]. The video clips are of 1920×1080 resolution and are captured from the vehicle traversing at a speed of $100km/h$. The detection on the Netherlands dataset is shown in Figure 5-4. Observing the input images and lane predicted images, the lane lines are in appropriate positions visually. This can be attributed to the good generalizability and robustness of the network. False negatives exist in the second image from the left in Figure 5-4. The third, fourth, and fifth lane prediction images in Figure 5-4 have incomplete predictions for the right lane. These shortcomings can be attributed to a different distribution of training data in the LLAMAS dataset in comparison to the Netherlands dataset. With the help of transfer learning methods [33], the network should be able to adapt to the Netherlands dataset very well.

Figure 5-4: Prediction of netherlands highway lanes using UNet spatio temporal attention network

5-2 Conclusion

In this thesis, an attempt to intuitively predict lanes in a sequence of images and reduce false positives is made. The current methodologies do not have the ability to focus on relevant images and important features in an image. This has been addressed by introducing spatio temporal attention networks.

To show the effectiveness of all proposed networks, five experiments were conducted. A baseline UNet network is extended with the attention module to create a UNet temporal attention network, UNet spatio temporal attention network, and UNet spatio temporal FC attention network. The proposed networks are trained on tvtLane and LLAMAS datasets and evaluated on tvtLane, TuSimple, LLAMAS, and the Netherlands dataset. Replacing the maxpooling layer with the convolution layer did not decrease the overall reduction in false positives. UNet spatio temporal attention network has an outstanding performance which is better than all networks in comparison. Increase in parameters is very small compared to the baseline network and other networks. The predicted lane images of the attention-based networks clearly show the importance of understanding the context in the image by the ability to focus.

5-3 Recommendation

Lane detection is a fundamental task for vehicles. Hence, a robust method which can adapt to all driving conditions, is adaptable and accurate is necessary. Towards this, some future recommendations are elaborated below. Recommendations include both the network architecture and datasets.

The proposed networks are trained and tested on an image sequence length of 5. Testing with higher number of images in the sequence can be beneficial in generalizability and improvement in accuracy of the network. Moreover, training the network with different intervals between the images in the sequence can simulate fast or slow driving conditions. The proposed network already can accept varying length sequences with different intervals between the images. It needs to be extensively tested for generalizability.

The attention modules proposed are implemented after the fourth downsampling layer and the first upsampling layer (Table 3-1). The attention layer can also be placed between different layers such as downsampling layers one and two, or downsampling layers two and three. Although the number of parameters of the network can increase due to the higher feature map sizes in these layers, the effectiveness of implementing the attention modules in these places should be tested.

Due to the network's architectural constraints, the upsampling layer heavily depends on the last image in the sequence. Sometimes the last image in the sequence may not have the relevant features needed to detect the lane lines. Hence, the network architecture should be

altered to upsample the image with the most relevant features.

The Netherlands dataset has been qualitatively assessed using UNet spatio temporal attention network. This dataset should be quantitatively assessed by labelling it. The network can also be tested on other lane detection datasets such as CULane, BDD100k to assess the robustness of the network.

Bibliography

[1] Mohamed Aly. Real time detection of lane markers in urban streets. In *2008 IEEE Intelligent Vehicles Symposium*, pages 7–12. IEEE, 2008.

[2] V. Kishore Ayyadevara. *Recurrent Neural Network*, pages 217–257. Apress, Berkeley, CA, 2018.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: A survey, 2014.

[6] Karsten Behrendt and Ryan Soussan. Unsupervised labeled lane marker dataset generation using maps. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[7] Amol Borkar, Monson Hayes, and Mark T Smith. Robust lane detection and tracking with ransac and kalman filter. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3261–3264. IEEE, 2009.

[8] Jingwei Cao, Chuanxue Song, Shixin Song, Feng Xiao, and Silun Peng. Lane detection algorithm for intelligent vehicles in complex road conditions and dynamic environments. *Sensors*, 19(14):3166, 2019.

[9] Cornelia Caragea and Vasant G Honavar. Machine learning in computational biology., 2009.

[10] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[12] TuSimple dataset. Tusimple dataset-lane detection. *TuSimple dataset lane detection*, 2017.

[13] Rahul Dey and Fathi M. Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600, 2017.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2019.

[17] Roger A Horn. The hadamard product. In *Proc. Symp. Appl. Math*, volume 40, pages 87–169, 1990.

[18] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1013–1021, 2019.

[19] SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.

[20] Zu Kim. Realtime lane tracking of curved local road. In *2006 ieee intelligent transportation systems conference*, pages 1149–1155. IEEE, 2006.

[21] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26, 2008.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Yeongmin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection. *arXiv preprint arXiv:2002.06604*, 2020.

[24] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[25] Jun Li, Xue Mei, Danil Prokhorov, and Dacheng Tao. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE transactions on neural networks and learning systems*, 28(3):690–703, 2016.

[26] Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, and Haowei Li. Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1394–1399. IEEE, 2020.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[28] Chan Yee Low, Hairi Zamzuri, and Saiful Amri Mazlan. Simple robust road lane detection algorithm. In *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, pages 1–4. IEEE, 2014.

[29] Haneet Singh Mahajan, Thomas Bradley, and Sudeep Pasricha. Application of systems theoretic process analysis to a lane keeping assist system. *Reliability Engineering & System Safety*, 167:177–183, 2017.

[30] Qi-Chao Mao, Hong-Mei Sun, Yan-Bo Liu, and Rui-Sheng Jia. Mini-yolov3: real-time object detector for embedded applications. *IEEE Access*, 7:133529–133538, 2019.

[31] John B McDonald and John B Mc Donald. Application of the hough transform to lane detection and following on high speed roads. In *in Motorway Driving Scenarios", in Proceeding of Irish Signals and Systems Conference*. Citeseer, 2001.

[32] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[33] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[34] 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*, 2018.

[35] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[36] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. *arXiv preprint arXiv:2004.11757*, 2020.

[37] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[38] European road safety observatory. Annual accident report 2018. *European road safety observatory*, 2018.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[40] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[41] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[42] Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, page 107623, 2020.

[43] Johan Vos, Haneen Farah, and Marjan Hagenzieker. Speed behaviour upon approaching freeway curves. *Accident Analysis & Prevention*, 159:106276, 2021.

[44] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[45] Wei Yang, Xiang Zhang, Qian Lei, Dengye Shen, Ping Xiao, and Yu Huang. Lane position detection based on long short-term memory (LSTM). *Sensors (Switzerland)*, 20(11), jun 2020.

[46] Seungwoo Yoo, Hee Seok Lee, Heesoo Myeong, Sungrack Yun, Hyoungwoo Park, Janghoon Cho, and Duck Hoon Kim. End-to-end lane marker detection via row-wise classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1006–1007, 2020.

[47] Jiyong Zhang, Tao Deng, Fei Yan, and Wenbo Liu. Lane detection model based on spatio-temporal network with double convgrus. *arXiv preprint arXiv:2008.03922*, 2020.

[48] 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, 2019.