

**THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY**

**ROAD LANE DETECTION SYSTEM WITH
CONVOLUTIONAL NEURAL NETWORK**

Master's Thesis

BORA TAŞHAN

İSTANBUL, 2017

THE REPUBLIC OF TURKEY

BAHÇEŞEHİR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

COMPUTER ENGINEERING

**ROAD LANE DETECTION SYSTEM WITH
CONVOLUTIONAL NEURAL NETWORK**

Master's Thesis

BORA TAŞHAN

Thesis Supervisor: ASST. PROF. TARKAN AYDIN

İSTANBUL, 2017

**THE REPUBLIC OF TURKEY
BAHÇEŞEHİR UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES COMPUTER ENGINEERING**

Name of the thesis: Road Lane Detection System with Convolutional
Neural Network

Name/Last Name Bora Taşhan

Date of the Defense of Thesis:

The thesis has been approved by the Graduate School of Natural And
Applied Sciences.

Prof. Dr. Nafiz ARICA
Graduate School Director
Signature

I certify that this thesis meets all the requirements as a thesis for the
degree of Master of Science.

Asst. Prof. Tarkan AYDIN
Program Coordinator
Signature

This is to certify that we have read this thesis and we find it fully
adequate in scope, quality and content, as a thesis for the degree of
Master of Arts.

Examining Committee Members

Signature

Thesis Supervisor

Asst. Prof. Tarkan AYDIN

Member

Asst. Prof. Cemal Okan ŞAKAR

Member

Asst. Prof. Görkem SERBES

ABSTRACT

ROAD LANE DETECTION SYSTEM WITH CONVOLUTIONAL NEURAL NETWORK

Bora Taşhan

Institute of Computer Science
Thesis Supervisor: Asst. Prof. Tarkan AYDIN

Mart 2017, 54 Pages

Occupying a significant place in today's automotive industry and being of vital importance in our daily lives, driving safety and driver assistance systems have become standard in more and more vehicles, especially by means of improvements in computer technologies and having a place in the industry, and they use more improved algorithms for more efficient solutions with developing technology equipment. As an example of these, being very important for the safety of driver, road lane detection system is now used in automotive industry and with the technological improvements, it has become more efficient and sophisticated. In the earlier stages, it was only a camera set in front of the vehicle and processing of the images taken by this camera. This innovation is followed by laser sensor, geographic coordinate system assistance and radar systems and with these innovations autonomous vehicle technology keeps improving.

In this thesis, by using deep learning methods, which are used for the processing of the data collected by internet technologies and are enabled to be applicable with the reduction of the costs of data storage components, memory units and multi-core microprocessors, one of the technique from deep learning is called convolutional artificial neural networks and computer vision methods is combined and autonomous vehicle road lane detection system method is proposed.

Keywords: Deep Learning, Convolutional Neural Networks, Lane Detection, Object Recognition, Machine Vision.

ÖZET

KONVOLÜSYONEL YAPAY SİNİR AĞLARI İLE ŞERİT TAKİP SİSTEMİ

Bora Taşhan

Bilgisayar Mühendisliği

Tez Danışmanı: Yard. Doç. Dr. Tarkan AYDIN

Mart 2017, 54 Sayfa

Günümüz dünyası otomotiv sektöründe önemli bir yer alan ve gündelik hayatımızda hayati öneme sahip olmaya başlamış sürüş güvenliği ve sürücü asistan sistemleri özellikle bilgisayar teknolojilerinin gelişmesi ve sektörde daha fazla yer edinmesi ile çok daha fazla araç içerisinde standart olmaya başlamış ve gelişen teknoloji ekipmanları ile birlikte daha verimli çözümler için daha gelişmiş algoritmalarından faydalanmaktadır. Bunlara örnek olarak verilebilecek ve bir sürücünün sürüş güvenliğinde önemli bir yeri olan şerit takip sistemleri otomotiv sektöründe kullanılmaya başlanmış ve yine teknolojinin gelişimi ile birlikte daha verimli ve komplike bir hale gelmiştir. İlk zamanlar araç önüne yerleştirilen bir kamera ve bu kameradan elde edilen görüntülerin işlenmesi ile başlayan bu yenilikleri lazer sensör kullanımı, coğrafi koordinat sistemi desteği ve radar sistemleri takip ederek otonom araç teknolojisinde gelişim devam etmektedir.

Bu tez içerisinde özellikle veri depolama elemanları, hafıza birimleri ve çoklu çekirdek destekli mikro işlemci maliyetlerinin düşmesi ile birlikte uygulanabilirliği sağlanan, inter- net teknolojileri ile toplanan veriler ve bu verilerin işlenmesi için gelişmeye başlayan derin öğrenme tekniklerinden konvolüsyonel yapay sinir ağları ve bilgisayar görüşü metodları birleştirilerek otonom araçlar şerit takip sistemi şerit algılama yöntemi önerilmiştir.

Anahtar Kelimeler: Derin Öğrenme, Konvolüsyonel Yapay Sinir Ağları, Şerit Algılama, Nesne Algılama, Makine Görüşü.

CONTENTS

TABLES.....	VI
FIGURES.....	VII
ABBREVIATIONS	IX
1. INTRODUCTION	1
1.1 AUTONOMOUS VEHICLES	2
1.2 LANE DETECTION SYSTEMS.....	3
2. LITERATURE REVIEW	5
2.1 LANE DETECTION SYSTEMS.....	5
2.2 FEATURE-BASED METHODS	5
2.3 CLASSIFICATION AND CONVOLUTIONAL NEURAL NETWORKS.....	11
2.3.1 Machine Learning.....	11
2.3.2 Neural Networks	13
3. METHODOLOGY	29
3.1 COLLECTING AND BUILDING DATASET.....	30
3.2 ESTABLISHING AND TRAINING CNN	32
3.3 PROCESSING ROAD IMAGE.....	37
3.4 OUTPUT	45
4. RESULTS	47
5. CONCLUSION	54
REFERENCES.....	55

TABLES

Table 3.1: Sample objects from training-set.....	31
Table 3.2: Numbers of training objects and labels.....	32
Table 4.1: CNN Training Parameters.....	47
Table 4.2: Evaluation and result of each parameter-set (28 classes).....	48
Table 4.3: Training images resized versions.....	50
Table 4.4: Evaluation and result of each parameter-set (2 classes).....	52
Table 4.5: Output of predictions (2 classes)	53



FIGURES

Figure 1.1: Vehicle Safety Sensor on Vehicle Windshield.....	1
Figure 1.2: A Typical lane detection system.....	3
Figure 1.3: Challenges of Lane Detection.....	4
Figure 2.1: Classification of Road Detection Approaches.....	6
Figure 2.2: Screenshot of RALPH.....	7
Figure 2.3: B-Snake Based Lane Model by Using 3 or 4 Control Points.....	8
Figure 2.4: Downward looking roadway departure warning system.....	8
Figure 2.5: IPM Sample. Left, input with ROI in red. Right, the IPM View.....	9
Figure 2.6: Robust lane detection sample.....	9
Figure 2.7: False detections of lane detection samples.....	10
Figure 2.8: Representation of biological neuron network.....	13
Figure 2.9: Biological neuron.....	14
Figure 2.10: Artificial neuron (Perceptron).....	16
Figure 2.11: Activation-output threshold relation in graphical form.....	17
Figure 2.12: A simple neural network.....	17
Figure 2.13: Sigmoid function.....	18
Figure 2.14: Hyperbolic tangent function.....	19
Figure 2.15: ReLU function.....	19
Figure 2.16: Fully-Connected artificial neural network.....	20
Figure 2.17: A Typical block diagram of CNN.....	22
Figure 2.18: 3D process of convolution.....	25
Figure 2.19: Representation of Max-Pooling and Average-Pooling.....	26
Figure 2.20: Representation of ReLU function.....	27
Figure 2.21: Fully connected layer processing.....	28
Figure 2.22: Convolution and fully connected layers.....	28
Figure 3.1: Flow of proposed method.....	29
Figure 3.2: A Sample image from Cal-Tech Cordova dataset.....	30
Figure 3.3: Sample road image from camera video.....	30
Figure 3.4: Architecture of LeNet.....	34
Figure 3.5: Training method flow.....	34

Figure 3.6: Plot of sigmoid function.....	36
Figure 3.7: Flow of image processing and classification.....	38
Figure 3.8: Samples from source road images.....	38
Figure 3.9: Greyscale color transformation on sample image.....	39
Figure 3.10: Input image, Edge detected image.....	41
Figure 3.11: Input image, processed image with dilation.....	42
Figure 3.12: Hough transformation visualization.....	43
Figure 3.13: Hough space and lanes on the source image.....	44
Figure 3.14: Lines are found on the given image by Hough line transformation...	44
Figure 3.15: Filtering out lines on the given image.....	45
Figure 3.16: Processed final image.....	46
Figure 4.1: Epoch/Accuracy ratio of 32x32 pixel size.....	49
Figure 4.2: Epoch/Accuracy ratio of 64x64 pixel size.....	49
Figure 4.3: Score function optimization of 32x32 pixel size.....	51
Figure 4.4: Score function optimization of 64x64 pixel size.....	52
Figure 4.5: Score function optimization of 64x64 pixel size for two classes.....	53

ABBREVIATIONS

AE	:	Auto encoder
ANN	:	Artificial Neural Network
BP	:	Backpropagation
CMC	:	Cumulative Matching Curve
CNN	:	Convolutional Neural Network
DML	:	Deep Metric Learning
FCM	:	Fuzzy C-Means
FNN	:	Feedforward Neural Network
GPS	:	Global Positioning System
GPU	:	Graphic Processing Unit
HOG	:	Histogram of Gradients
HSI	:	Hue, Saturation, Intensity
HSV	:	Hue Saturation Value
ILSVRC	:	ImageNet Large Scale Visual Recognition Challenge
IMM	:	Interacting Multiple Models
IPM	:	Inverse Perspective Mapping
KNN	:	k-Nearest Neighbor
LDW	:	Lane Departure Warning System
LIDAR	:	Laser Image Detection and Ranging
MAE	:	Mean Absolute Error
ML	:	Machine Learning
MLP	:	Multi-Layer Perceptron
MSE	:	Mean Squared Error
PNG	:	Portable Networks Graphics
POI	:	Point of Interest
PR	:	Pattern Recognition
RANSAC	:	Random Sample Consensus
RGB	:	Red, Green, Blue
RNN	:	Recurrent Neural Network
ROI	:	Region of Interest
SGD	:	Stochastic Gradient Descent

SIMD : Single Instruction Multiple Data
SVM : Support Vector Machines
ZIP : Archive file format supports lossless data compression.



1. INTRODUCTION

Having a significant place in today's world, automotive industry has become a part of our lives and with this improvement, in order to provide safety of life and for a more secure drive in a car which is one of the products of this industry, it is very important to equip necessary instruments. According to the study based on data from Turkish Statistical Institute and General Directorate of Security, in Turkey, 2015, from 1,313,359 accidents in total, 304,421 accidents caused fatal injuries and in these accidents, percentage of driver deaths is 40,7 (Turkstat 2015). Percentage of 89,3 of these accidents are caused by driver's fault, ranking first. Because of that, driver assistance systems, a part of autonomous vehicle concept in automotive industry, stand out especially for life safety and safe drive. Lane departure warning system, a part of related drive assistance systems, is offered as a part of vehicles.

In 2009, in the USA, National Highway Traffic Safety Administration carried out a work and in the following years many vehicle manufacturers release lane detection system and related preventive systems with their cars (NHTSA 2005). In Figure 1.1 lane detection camera in Volvo S60 can be seen.

Figure 1.1: Vehicle Safety Sensor on Vehicle Windshield



After lane departure systems were offered in Cima released by Nissan Motors in 2000s, many leading car manufacturers also started offering this system. With the data taken from lane tracking system, both audio alarm systems and seat vibration systems warn the driver and lane detection systems become a part of our lives as a preventative system.

Collaterally, automotive industry developed a lane tracking system, with different methods with the images taken from a camera inside the vehicle, tracking the road and by using these lane tracking system methods, they produce preventative safe drive solutions as lane departure warning.

These systems both help drivers and play a key role in developing autonomous car i.e. intelligent vehicle.

1.1 AUTONOMOUS VEHICLES

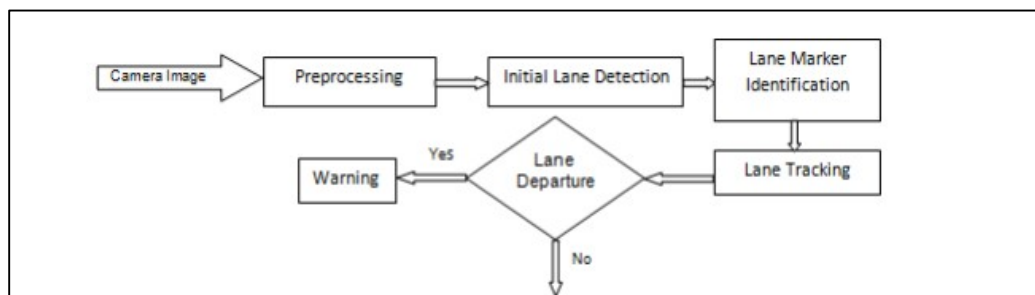
Autonomous car or intelligent vehicle is a vehicle that is capable of sensing external environment by itself and navigating without the help of driver or without any human interference. As a result of the studies on autonomous car, date back to 1920s, the first working model of the car was appeared in 1980s. Carnegie Mellon University's NAVLAB and ALV (Jochem et al. 1995) projects in 1984 were followed by Eureka Prometheus project of Mercedes-Benz and Bundeswehr University Munich in 1987 (Dickmanns 2002). From these vehicles navigating without any human interference, NAVLAB-5, project of Carnegie Mellon University, completes 2,797 miles from Pittsburgh to San Diego with approximately 102,3 km/h average speed by itself (Pomerleau and Jochem, 1996). Nowadays, these vehicles started to take a significant part in our lives with the developments of Google, Tesla and similar manufacturers and lead the way for using many preventative methods and prevention of traffic accidents with more effective road use and more secure drive. GPS, Laser, Odometer and Computer vision techniques are used to develop self-driving skills and lane detection system takes an important role in safe drive.

1.2 LANE DETECTION SYSTEMS

With the heavy increase in car traffic, especially 30% of the accidents on highway are caused by lane changing and most of these accidents happen because the driver is exhausted or absent-minded. Therefore, the systems developed for driver changing lanes accidentally or not missing the lane, not only prevent many traffic accidents but also save many lives. These driver safety and preventative systems are called Advanced Driver Assistance Systems (ADAS). Some example part of ADAS systems are night vision, cruise control for drivers, blind spot detection, traffic light detection and control system. Lane detection system is also a part of ADAS. The purpose of these lane detection systems is to detect lanes during driving, informing driver assistance systems of lanes and ensure the system gives a warning in case the vehicle leaves its lane.

In intelligent vehicle systems, the vehicle works coordinately with these infrastructure systems and aims to have a more secure drive and traffic. Basically, lane detection systems show the lanes to the driver on a screen in the vehicle, but more developed systems analyze lanes, other vehicles on the road and whether it is precise time to change lanes and warn the driver. Lane detection systems use camera, laser, LIDAR and GPS technologies for these processes (Borkar et al. 2011). In Figure 1.2 a typical lane detection system flow can be seen.

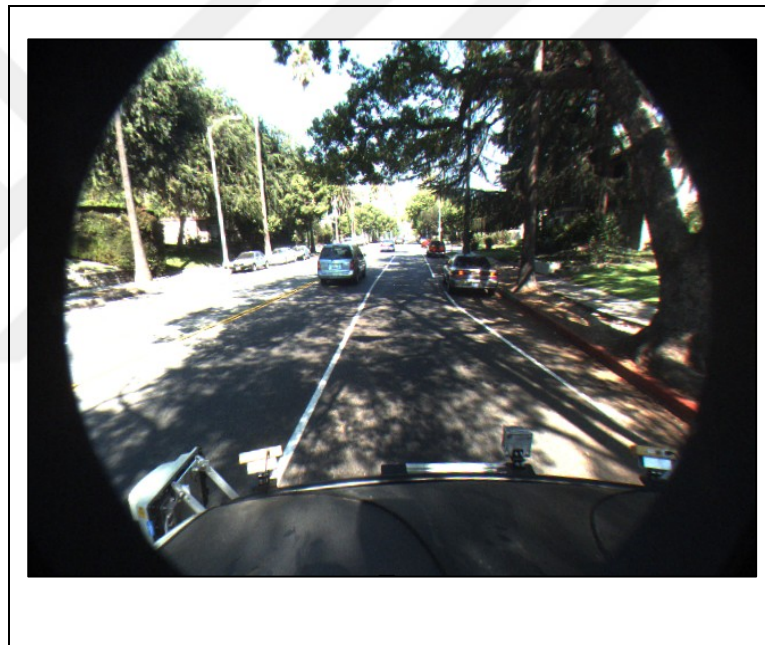
Figure 1.2: A Typical lane detection system



Source: Pallavi V. Ingale, Prof. K. S. Bhagat, “Comparative Study of Lane Detection Techniques”

In many proposed systems, lane detection systems locate primitive objects (e.g. lane, vehicle, road limits) such as predefined vehicles, signboards and road surface markings. However, at this stage, several difficulties caused by environmental conditions lead these algorithms to have some problems. In general, these difficulties are as follows; vehicles parked or on the move, erased or worn out road lines, shadows, non-standard road markings and lanes, non-standard curved lines. Intersecting road signs and road surface markings. In Figure 1.3, there is a shadowy road image.

Figure 1.3: Challenges of Lane Detection



In order to overcome these problems and develop much more reliable lane detection systems, there are still research works going on and many proposals are made (Kim 2008).

2. LITERATURE REVIEW

2.1 LANE DETECTION SYSTEMS

There are many approaches in lane detection such as feature-based or model-based (Mistry and Makwana 2014).

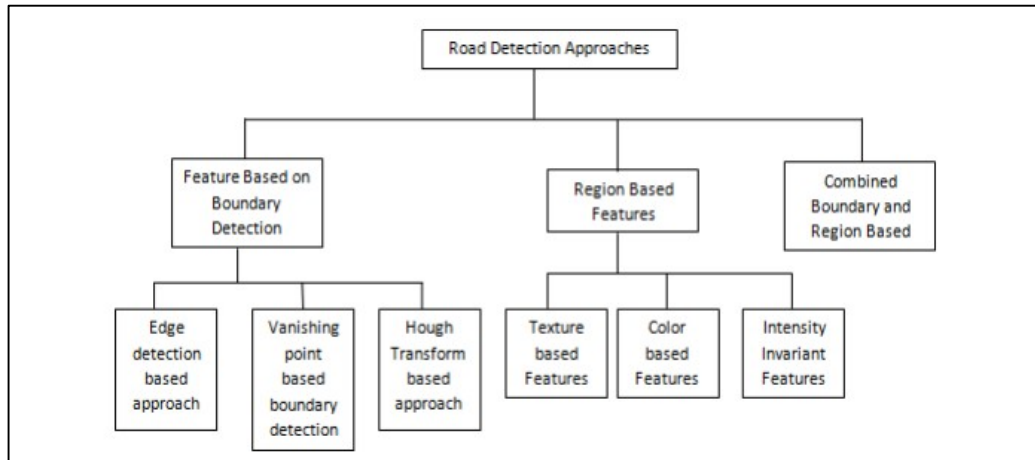
2.2 Feature-based Methods

Feature-based methods find and track low level features such as lanes on road, side lines of lanes. However, the success of feature-based approach depends on how apparent lines are. Consequently, lines which are less or not apparent are affected by noise and occlusions on camera image and these conditions diminish the possibility of success of this approach.

Model-based Methods

Model-based methods define road lanes as a sort of curve model and several significant geometric parameters on this model. Model-based approach is more resistant to poor lane images and noises and shows more success in comparison to feature-based approach. However, when model-based approaches are built in accordance with certain scenes, one method might be successful in one scene but other might not, so it makes the approach less adaptive. Moreover, the learning algorithm for learning parameter which will be built on modelling, is more time-consuming in proportion to other approach (Han and Hahn 2010). In Figure 2.1, classification of road detection approaches is seen.

Figure 2.1: Classification of Road Detection Approaches



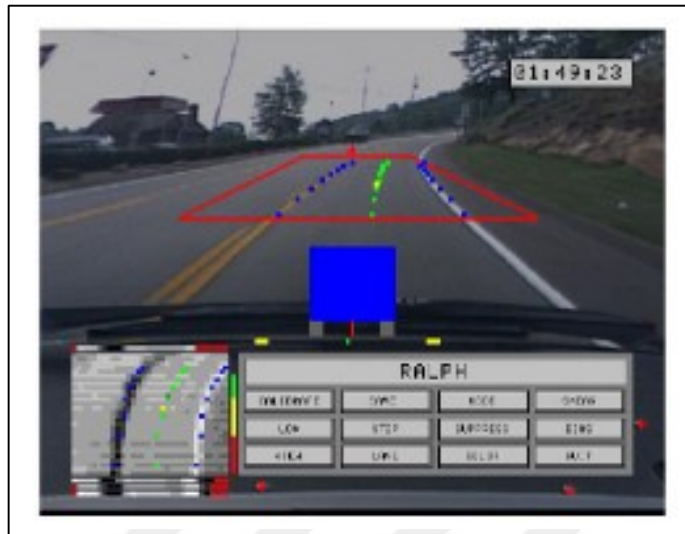
Source : Pallavi V. Ingale, Prof. K. S. Bhagat, “*Comparative Study of Lane Detection Techniques*”

Researches and Proposes About Lane Detection Systems

Lane detection approaches and solutions developed so far are explained in literature review below.

In their article published in 1996, Dr. Dean Pomerleau and Dr. Todd Jochem proposed the system that ensures the vehicle is tracking the related lane by the help of processing images taken from the camera on the vehicle and calculating lane offsets with vision system they called RALPH (Rapidly Adapting Lateral Position Handler) (Pomerleau and Jochem 1996). After the studies, tests performed on an approximate 2850-mile road succeeded. Although RALPH system succeeded in tests, it was affected when lanes could not be seen for heavy traffic or reflections on daylight.

Figure 2.2: Screenshot of RALPH.



Source : D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering,"

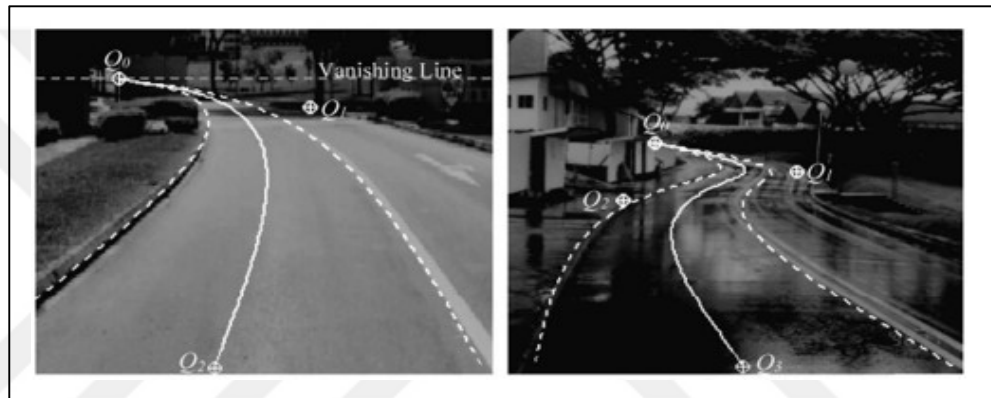
B.M. Broggi proposed a method called GOLD (Real-time Stereo Vision Parallel System for Generic Obstacle and Lane Detection) system which converts the given image it got from the in-car camera into a bird's-eye view image on a new image (Inverse Perspective Mapping) and made lane lines become almost vertical lines and detect lines making them quasi-vertical objects on a dark background (Broggi 1998).

According to the article Kreucher and Lakshmanan published in 1998, by an algorithm named LOIS (Likelihood of Image Shape), lanes could be detected regardless of problems caused by shadows, lanes blocked by other vehicles (occlusion) or different light environment (Kreucher and Lakshmanan 1998). By LOIS algorithm, all possible lane objects on the road are identified with parametric of shapes. Then it is determined whether the lines on the road are lane lines or not by comparing lane objects and parametric lane data. LOIS (for Likelihood of Image Shape) uses a deformable template approach. Family of parametric shapes describes the all possible ways that the lane edges could appear in the given road image. A function is defined that its value is proportional to how accurate set of lane shape parameters matches the pixel data in a given image. The lane detection

process is performed by finding the lane shape parameters which maximize the function for the given specified image.

In the article Y.Wang and his friends published in 2004, with B-Snake spline method, set the road as a geometric model and in order to do it, they used CHEVP (Canny and Hough Estimation of Vanishing Points) and extracted geometric model parameters (Wang et al. 2004). Related algorithm became highly successful, especially where shadows are confused with lane data. However, this algorithm was affected by the shadows of poles or tree trunks and was not a solution for these shadows intersecting lanes.

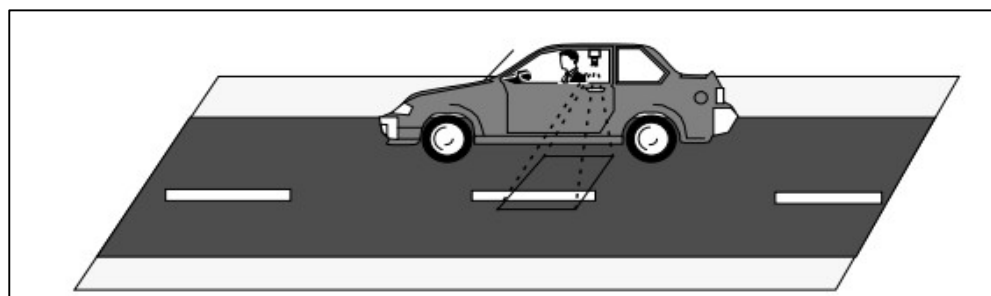
Figure 2.3: B-Snake Based Lane Model by Using 3 or 4 Control Points



Source: Y. Wang et al. Lane detection and tracking using B Snake”, In: Image and Vision Computing 22, pp: 269-28, 2004.

In an article published in 2004, Chen and his friends placed a system called AURORA next to the vehicle and could detect lane lines using wide-angle colorful camera image (Chen et al. 2004).

Figure 2.4: Downward looking roadway departure warning system



Source: M. Chen., T. Jochem and D. T. Pomerleau, “AURORA: A Vision-Based Roadway Departure Warning System”

Jung and his friends could detect lane lines by using edge detection method with squares angular approximation as published in an article in 2005 (Jung et al. 2005). In his article published in 2008, Mohamed Aly proposed a real-time, robust and efficient lane detection algorithm. The algorithm takes the image from the camera mounted on the vehicle and creates ROI (Region of interest) around the road section from the image. After this ROI operation, it applies IPM (Inverse Perspective Mapping) in order to distribute all information of the region homogenously on the given image and it applies selective Gaussian filter for bringing out the lanes on the road image (Aly 2008). After applying Hough transform, it detected vertical lines on that area and following RANSAC (Random Sample Consensus) line fitting and RANSAC spline fitting processes, detected the places of the lanes on the image. In Figure 2.5, the ROI and IPM result is seen.

Figure 2.5: IPM Sample. Left, input with ROI. Right, the IPM



Source: M. Aly, "Real time Detection of Lane Markers in Urban Streets", In *IEEE Intelligent Vehicles Symposium*

Algorithm was resulted very good performance on many conditions. In Figure 2.6, the result images of the given road data is seen.

Figure 2.6: Robust lane detection sample



Source: M. Aly, "Real time Detection of Lane Markers in Urban Streets"

Although the algorithm works stably and successfully in many environments, it is affected by crosswalk lines and other signs. In Figure 2.6. False detections of lane detection samples are seen.

Figure 2.7: False detections of lane detection samples



Source: M. Aly, “Real time Detection of Lane Markers in Urban Streets”

In his article published in 2008, Z.Kim proposed a stable algorithm for unexpected signs, non-standard curved lanes and non-standard lane changes on the road. Related algorithm develops hypothesis using random sample consensus and particle filtering algorithms (Kim 2008). O.O. Khalife, in his article published in 2009, processed the video frames taken on road in real time with the help of the camera he placed on the vehicle and could detect road lanes regardless of light and shadow changes (Khalifa and Hashim 2009). The algorithm converts the image taken from the camera into a Greyscale image. After it applies noise reduction, with canny edge detection and Hough transformation, it detects right and left lanes. The algorithm successfully processed the real time video frame with sufficient speed, however it was not very successful in detecting sharp curves in shadowed areas. F. Mariut 2012 proposed a method that detects the lane markers, characteristics of them and is able to detect direction of travelling. The Hough Transform was used to detect the lines in images. A technique was developed for being ensure to right detection of lane mark by extracting inner margin of the lane (Mariut et al. 2012).

In 2006, Sun and his friends detected lanes by using HSI color model (Sun et al. 2006). Although pixel values of road lanes are different from the pixel values of other areas, even in the studies performed with RGB color model used in image processing, in HSI color model, pixel values in lane area are prominently different from pixel values of other areas. Therefore, it makes it easier to detect the lanes on HSI color model. Sun and his friends converted the image taken from camera

in RGB format into HSI format and extracted the lower part of the image to detect lane lines. Because the intensity value of lane line pixels is explicitly different from other pixels' value, this intensity value is used as threshold and divided into separate clusters by Fuzzy C-Means method. Then the image is converted into binary image to detect shapes and among these shapes, the ones that have a particular width and length rate are assumed to be lanes. Kim and his friends had a similar approach in the article they published in 2012. After they had binarized 640x480, 24-bit road image, they detected the frame of the object using 4-directional contour tracking algorithm and in every 50 pixel, they vectorised the object and extract its characteristic. They detected the lanes on the image by dividing the extracted vectors into clusters with the help of FCM (Kim et al. 2012).

2.3 CLASSIFICATION and CONVOLUTIONAL NEURAL NETWORKS

2.3.1 Machine Learning

Tom M. Mitchell quoted a formal definition of machine learning as (1997, p. 2)

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E .

Even though it is most often used interchangeably with the term Pattern Recognition (PR), they are not the exactly same and machine learning is evolved from Pattern Recognition and computational learning theory. Machine learning is study of constructing and exploring algorithm and model from given data for making data driven prediction or decision rather than using strict and static algorithms to apply prediction on data set such as predicting price of real estate, deciding whether given human image is male or female, classification, deciding whether email is spam or image classification. In order to apply prediction or decision, machine learning techniques creates model with parameters and learns parameters by optimizing them by using training data or past experience.

Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process (Goodfellow, Bengio and Courville p.104). In supervised learning, there is training set $X = \{x, r\}$ where x the feature is or pattern vector, r is the desired output which is called label or target and t is the index of sample in the dataset T , N is the feature count of X . The aim of supervised learning is learning a mapping from input x to an output r to use it for estimating or predicting accurately \hat{y} of given value \hat{x} where \hat{y} and \hat{x} is not the element of X . There are many supervised learning techniques in the literature which are k nearest neighbors (k-NN), decision tree, multilayer perceptron (MLP), support vector machine (SVM), artificial neural networks (ANN), linear regression and so on. Since supervised learning is provided by the output y by instructor or teacher in order to show machine learning algorithm what to do, in contrary of supervised learning, unsupervised learning algorithms experience a dataset X which contains input data x and no information about target y . Unsupervised learning aims finding or extracting useful patterns of unlabeled dataset X without instructor or given information about data. Many techniques such as clustering, anomaly detection widely use unsupervised learning algorithm.

Classification and regression, the two important applications of machine learning and supervised learning algorithm, are mainly employed to solve such problems. Classification is the method that finding the category or label of given feature such as deciding whether the given image of animal is cat or dog. In contrast to classification, regression mainly deals with continuous values of data such as predicting sales price of used bike. Basically, in machine learning the regular approach is that constructing a model $(x|\theta)$ where x is the input vector and θ are the parameters (Alpaydm, p 39-42). Main goal of the machine learning algorithm is optimizing θ by minimizing loss E over the each data in dataset X

$$E(\theta|X) = L(r' - g(x|\theta)) \quad (2.1)$$

Where $L(.)$ is the loss between prediction r' and model $g(x|\theta)$ which utilizes current parameter values of θ .

$$\theta'' = \mathit{arg\ min}_{\theta} E(\theta|X) \quad (2.2)$$

After the parameters θ'' is found which is called learning phase, the model with parameters constructed is used for testing each individual feature of test dataset which is different than training set and contains labeled data depicting the accuracy of model by using MAE or MSE in order to measure performance of the model.

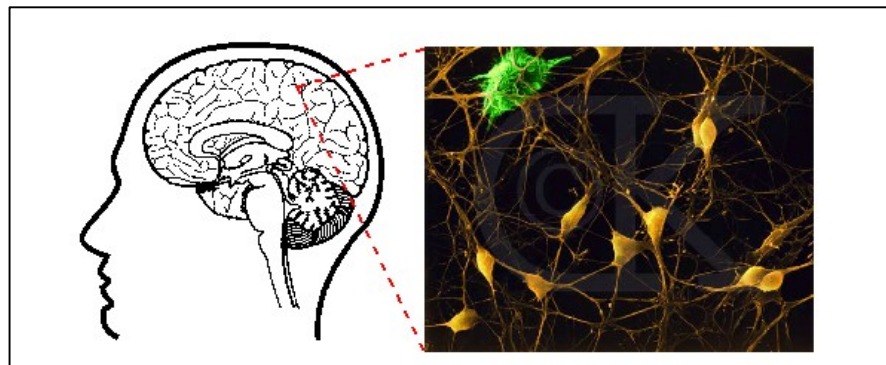
$$MAE = \frac{1}{n} \sum_{i=1}^n |f_i - y_i| = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (2.3)$$

2.3.2 Neural Networks

Biological Inspiration

The human brain is approximately composed of about 86 billion neurons where each neuron is connected to about 10,000 other neurons according to researchers.

Figure 2.8: Representation of biological neuron network

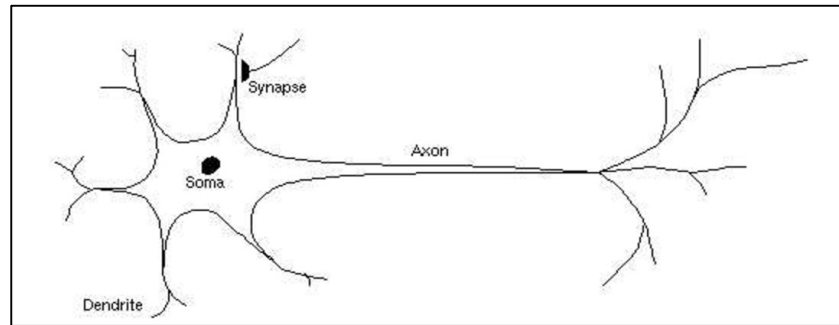


Source: <https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Biology/index.html>

A neuron is composed of soma (body), dendrites and axons which are input and output channels and connect neurons to each other. Each neuron receives electrochemical signals/inputs from other neurons from the dendrites. When the sum

of these electrochemical inputs is enough powerful to activate it, the neuron transmits the signal along the axon and passes this electro-mechanical signal to next neurons which are connected to axons. Those attached neurons may fire then.

Figure 2.9: Biological neuron



Source:<https://cs.stanford.edu/people/eroberts/courses/soco/projects/neuralnetworks/Neuron/index.html>

The important point is that a neuron fires only when the total signal received at the body exceeding a certain level which means the neuron either fires or do not fire. Our entire brain is composed of these interconnected electrochemical transmission neurons where very large of these simple units manage to perform very complex task. This biological model is the base of artificial neural networks, however artificial neural networks still do not come close to complex model of the brain, and artificial neural networks have shown their ability that they are good in some problems but those problems are still very easy for human brain to solve such as image recognition where human brain does not need training process like ANNs do.

Artificial Neural Network

Warren McCulloch and Walter Pitts created a computational model for neural networks (McCulloch and Pitts 1943). The model was showing two distinct approach, one focused on biological process in the brain and another focused on the artificial intelligence applications of neural networks. In 1949, Donald Hebb mentioned in his book and pointed out that the connections between the neurons that fire at the same time are enhanced which is essential for human brain learning (Hebb 1949, p. 62). Frank Rosenblatt (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer computer learning network using

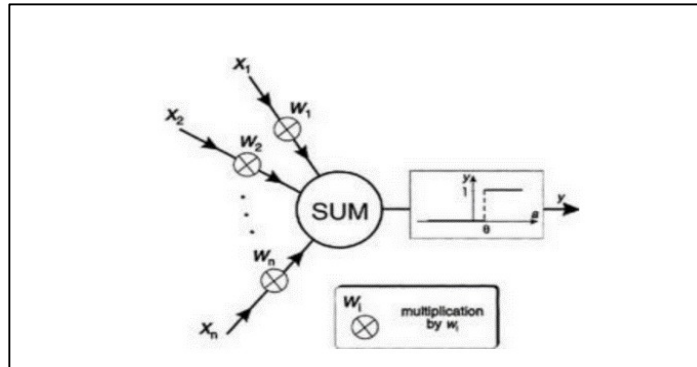
simple addition and subtraction. With mathematical notation, Rosenblatt also described circuitry not in the basic perceptron, such as the exclusive-or circuit, a circuit which could not be processed by neural networks until after the backpropagation algorithm was created by Paul Werbos (1975).

However, since neural network algorithm needed computational power and resources, many other approaches in AI like Support Vector Machines took the place of the study, therefore improvements and works on neural networks were quite silent. Eventually, neural networks with deep layered networks have become popular after 2000 due to dramatically improved computation resources and parallelism of computers.

Perceptron

Perceptron is the main computational unit and the mathematical model of the biological neuron. While in actual neurons the dendrite receives a signal from the axons of other neurons, in the perceptron those electrochemical signals represented as binary or numerical values. In actual neurons, between the dendrite and axons, signals are modulated in various amounts where the perceptron modeled it by multiplying each input value by a value called the weight. The neuron fires an output signal only when the total strengths of the input signals exceed a certain threshold, and the perceptron similarly accomplishes it by calculating weighted sum of the inputs to present is total strength of the input signals and applying a step function on the sum to determine output where it fires other neurons which are connected. According to its mathematical modeling, a perceptron is composed of several binary inputs ($x_1, x_2, x_3 \dots x_n$), Weights ($w_1, w_2, w_3 \dots w_n$) a real number expresses the importance of each input values.

Figure 2.10: Artificial neuron (Perceptron)



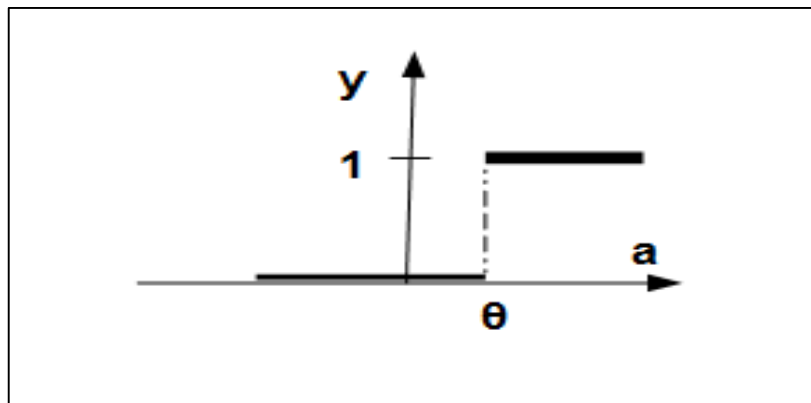
Source: An Introduction to Neural Networks (Gurney 1997, p. 30)

When input values are received, perceptron calculates $\alpha = \sum_{i=1}^n w_i x_i$ in order to obtain an activation output value to determine if it's less or greater than some threshold value θ (theta).

$$\text{output} = \begin{cases} 0, & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1, & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases} \quad (2.4)$$

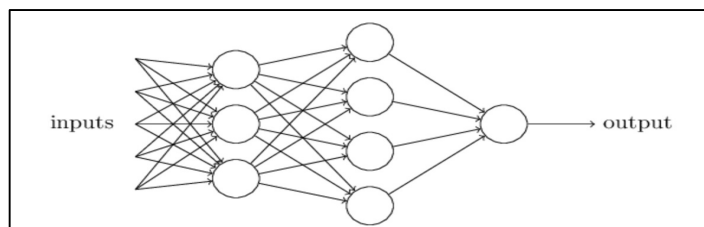
When this activation value exceeds (or is equal to) (theta) then the perceptron outputs a “1” (action potential) , and if it is 0 or less than Q (theta) then it emits “0”. This might be represented graphically as shown in figure X.Y where the output has been designated by the symbol y. This relation is called “step function” and it decides whether the perceptron should fire if the activation exceeds the threshold. In Figure 2.10, graphical representation of the function is seen.

Figure 2.11: Activation-output threshold relation in graphical form



Obviously, the perceptron alone is not a model of complete human brain and how it makes decision. But connecting such perceptron's to each other and creating a network with them can perform some decision on some sort of problems. In Figure 2.11, a simple connected neuron called neural network is seen.

Figure 2.12: A simple neural network

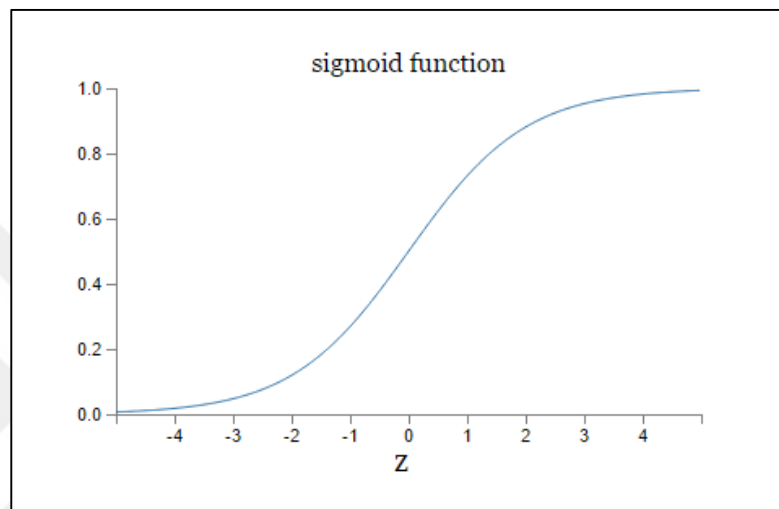


The network is composed of perceptron's and uses step functions as activation function and accepts binary output and produces binary output which could be utilized to solve basic classification problems. However, when it comes to learning phase, changing weights according to some minor changes could flip neuron output from 0 to 1 or vice versa and it could change network behavior completely. To overcome this problem, another type of neuron called sigmoid neuron could be used. Just like perceptron, sigmoid neurons ($x_1, x_2, x_3 \dots x_n$) and these inputs can take any values between 0 and 1. Similar to perceptron, sigmoid neuron has weights for each input ($w_1, w_2, w_3 \dots w_n$) and an overall bias, b . Instead of output is 0 or 1 like perceptron, sigmoid produces output between 0 and 1 which is $\sigma w \cdot x + b$ where σ is called the sigmoid function or logistic function and defined b

$$\sigma(\mathbf{z}) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\sum_j w_j x_j - b)}} \quad (2.5)$$

Suppose that $z \equiv w \cdot x + b$ is large positive number then $e^{-z} \approx 0$ and $\sigma(z) \approx 1$ which means when $z \equiv w \cdot x + b$ is large positive the output of sigmoid neuron approaches to 1 and when it input is very negative then $e^{-z} \rightarrow \infty$ and $\sigma(z) \approx 0$.

Figure 2.13: Sigmoid function

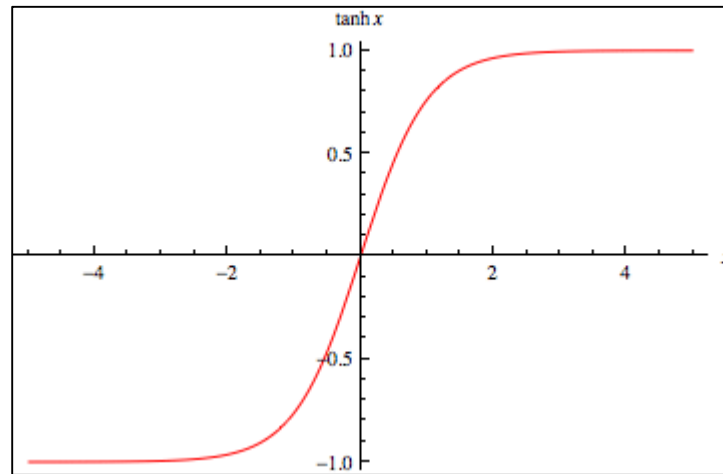


Since sigmoid function produces input between 0 and 1 and overcomes to binary output of step function, there are also another activation functions such as hyperbolic tangent, Rectified Linear Unit and Softmax.

The Hyperbolic Tangent is very similar to sigmoid function but it takes values between -1 and 1 and defined by

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.6)$$

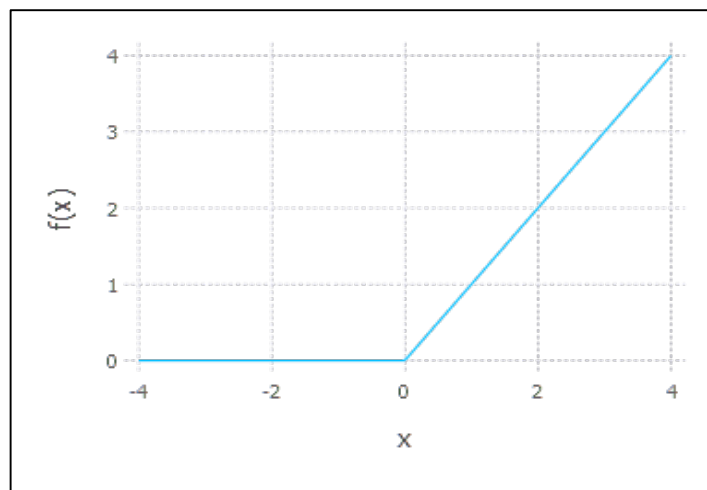
Figure 2.14: Hyperbolic tangent function.



The Rectified Linear Unit (ReLU) is another activation function where it is mainly used in deep neural networks. In comparison to sigmoid function, ReLU activation function take input values from $[0$ to $\infty)$ and defined by

$$f(x) = \max(x, 0) \quad (2.7)$$

Figure 2.15: ReLU function.



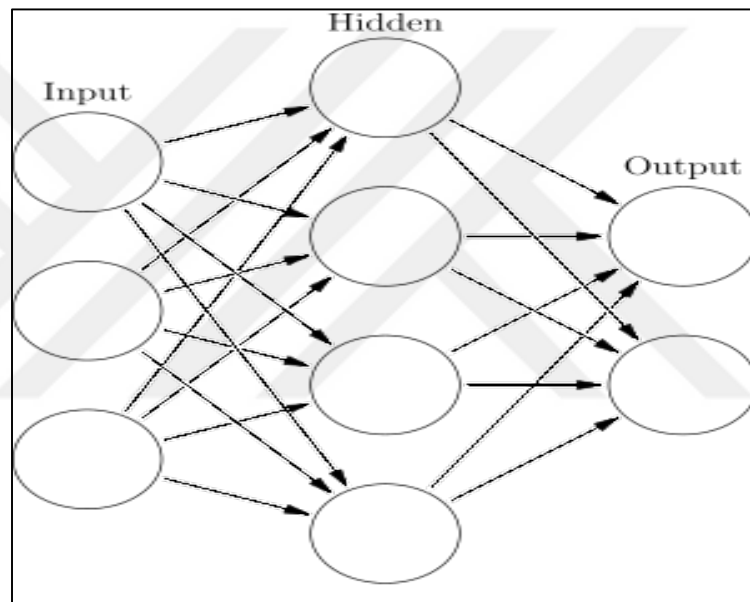
The Softmax function is mainly used as normalizing function commonly used in last layer of neural networks in order to encode probabilities output vector for classification tasks and defined by

$$y_i = \frac{e^{x_i}}{\sum_j^N e^{x_j}} \quad (2.8)$$

The Architecture of Neural Network

Since an artificial neuron is not sufficient to model the brain, artificial neuron could be connected under a network architecture as it is presented in figure 2.15 with fully connected layers.

Figure 2.16: Fully-Connected artificial neural network.



The network architecture's first layer is input layer composed of input neurons, the last layer is output layer and between the input and output layer is hidden layer. Hidden layer might have one or multiple layers according to architecture. Mostly multi layers networks are called MLP (Multi-layer perceptron). Input signals are processed in neurons and propagated to next neuron's inputs. As for patterns of connections, the main distinction of network topologies are feed-forwarded networks (FFN) and recurrent networks (RNN).

In Feed-Forward networks, the data flow from input to output is feed forwarded. The data processing can be extend to multiple layers of units but no feedback connections there between each layer which means that data propagated from

layers to inputs of next layer without a form of cycle. Contrary to feed-forward networks, recurrent neural network connections have form of cycle between units to do feedback. The main property of RNN is having dynamical properties which causes network to have temporary memory to process input sequences.

Backpropagation

To establish a well accurate mapping between input and output of a neural network, a common method is to train network parameters (weights) with supervised learning. This process requires to have a training dataset with output data for given input data. By this training data, a network can compare its output with actual output. Suppose the desired output is y and actual output is \hat{y} is passed to a Cost or loss function C which minimize the output by adjusting the weights and biases of the network. Let θ to be the set of all parameters of the network and then the training of the network by supervised learning is to minimize,

$$\min_{\theta} \frac{1}{N} \sum_{t=1}^N C(y_t, \hat{y}_t | \theta) \quad (2.9)$$

The process is passing data through the network, calculating the cost and readjusting the parameters until the network reaches to sufficient accuracy when the training set is validated by the test set. This method is called backpropagation and with gradient descent, backpropagation propagates the gradients of the cost function with parameters and puts back them to network.

Stochastic Gradient Descents

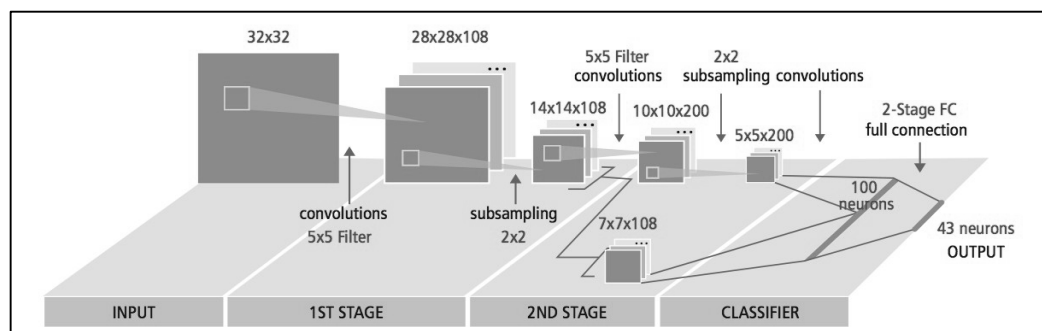
In order to update parameters to reach for acceptable accuracy, a couple of methods can be employed such as Random search, Random local search or Gradient Descent. Random search is very inefficient approach because of finding new parameters randomly and following continuously which parameters produce the best result in every iteration. Random local search produces better result where

iteration starts with random W and generates random perturbations δW to it and if the loss is better, it updates the parameters and keeps continuing iterations.

Convolutional Neural Networks

CNN uses different approach and has special methods similar to multilayer neural networks which means it might be accepted as special version of multilayer neural networks. A convolutional neural networks is composed of one or more convolutional layers which is together with subsampling/pooling layer and at the end is followed by fully connected layers which is seen in neural networks. The idea behind the convolutional neural networks is motivated by visual perception which is called visual cortex of human brain. The visual cortex is composed by many cells which are responsible for detecting light in small and sub-regions of the receptive fields which is a visual field where more complex cells have got larger receptive fields. These receptive field cell process as local filters on the input space where they detects edges in the given input space. These results of researches have motivated and lead CNN convolutional layers to perform the same functionality that cells do on visual cortex. Recognizing an object in an image by CNN is shown in Figure 2.17. Features of a layer is fed by input from a set of features located in a small neighborhood in the previous layer which is local receptive field. The aim of these local receptive fields is extracting elementary visual features such as edges, horizontal lines, corners in order to combine them with higher layers.

Figure 2.17: A Typical block diagram of CNN.



Source : Hijazi, S., Kumar, R., & Rowen, C., 2015. Using Convolutional Neural Networks for Image Recognition.

CNN is used in many areas such as pattern recognition, image recognition, natural language processing and speech recognition. Since CNN does not need any hand-designed feature extraction process which is common in typical pattern or image recognition feature extraction process, it has become main advantage of the CNN usage in many areas. In convolutional neural networks, weights of the convolutional layer are being used for extraction of features and the fully connected layer at the end is used for classification. This classification process is determined during the training process. Also a CNN network structure can be improved in order to save memory requirements and computational complexity as well as giving better performance for different kind of problems such as image or speech recognition.

CNN's Components

Convolutional neural networks are composed of multiple stacked layers in order to establish a complex architecture for classification problems. These layers are divided into four different type of layers which are convolutional layers, pooling/subsampling layers, non-linear layers and fully connected layers. Typically, the purposes behind of these layers are to reduce the dimensions of intermediate layers, reshaping and simulating fully connected layers. Following sections describe these components that are given.

Convolutional Layers

The convolution is a 2D operation which is defined by kernel of size $k \times k$. Given an input image X with $N \times M$, the convolutional kernel is sliding from left-to-right and top-to-bottom along the image X and examining kernel with surrounding pixel values and put the result to output image Y where the output pixel location i, j is calculated

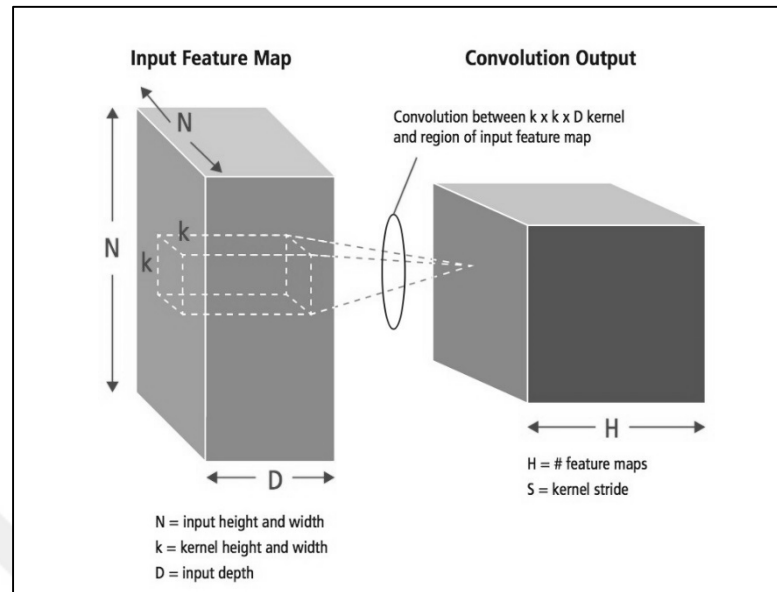
$$Y_{ij} = \sum_{i'=1}^k \sum_{y'=1}^k k_{ij'} X_{i-\frac{k+1}{2}+i', j-\frac{k+1}{2}+j'} \quad (2.10)$$

This convolution operator is denoted by $*$ and thus

$$Y = k * X \quad (2.11)$$

By using this convolution operator, this operation is the main actor of the convolutional layer where it finds and extracts different features of the given input space N . Convolution layers extract low-level features such as edges, lines or corners. Higher level layers extract higher-level features. Figure 2.18 shows the process of 3D convolution used in CNN. Since convolution is a 2D process, the image which is composed of R, G and B channels are divided into separate given input. The input is size of $N \times M \times D$ and is convolved with kernel H where each size of kernel is $k \times k \times D$. Convolution operation is fed by input with kernel and to produce output feature, with H kernels it produces H independent features. It starts from top-left corner of the specified input and moved to right S element by moving from left to right where S is called “Stride” defines number of step. When it reaches to the top-right corner, the kernel is moved to downward direction by one element and it repeats to move from left to right by the same stride. This process is kept continue until the bottom-right corner is reached by kernel. For example, for the case $N, M = 48, k = 5, S = 1$, there are both 44 different positions for left to right top to bottom that the kernel can process which can be formulized by $(N - k + 1) \times (M - k + 1)$. Each feature in the output will contain 44x44 elements corresponds to these positions. The kernel is processing under sliding window technique for each positions. In this process, $k \times k \times D$ elements of input and $k \times k \times D$ elements of kernel are multiplied element-by-element and accumulated. Therefore, $k \times k \times D$ multiply-accumulate operations are used for creating one element of one output feature.

Figure 2.18: 3D process of convolution.



Source : Hijazi, S., Kumar, R., & Rowen, C., 2015. Using Convolutional Neural Networks for Image Recognition.

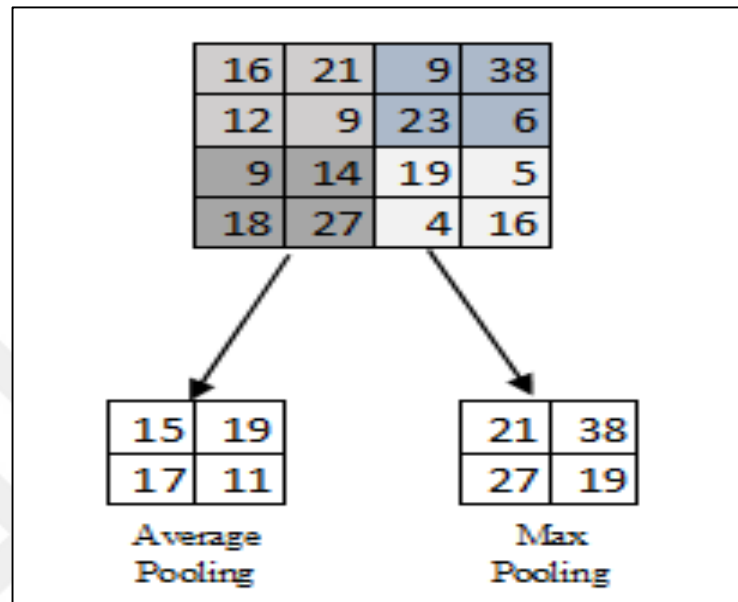
Pooling/Subsampling Layers

The aim of the layer is reducing the spatial size representation dimension size to reduce the number of parameters and computation in the network. In this layer, two different method is seen to apply pooling. One is max pooling and other is average pooling. Both method does the same process on the given input space by divide it into non-overlapping 2D spaces. In Figure 2.17, layer in second stage is the pooling/subsampling layer where every input feature contains of 28x28 dimension and is divided into 14x14 sub-regions of size 2x2. The methods of calculations are average pooling calculates the average of 4 values of 2x2 sub region where max pooling calculates the max value of again given 4 values of 2x2 sub region.

Figure 2.19 represents the pooling process where the input is size of 4x4 and 2x2 sub- sampling. This 4x4 image is divided into four different matrices of size 2x2. Max-pooling method produces the maximum value of the four values in the 2x2 matrix is the output. However, average pooling method produces the average of

the four values is the output. In case of average pooling, if the output value is calculated to fraction, it is rounded to nearest integer.

Figure 2.19: Representation of Max- Pooling and Average-Pooling



Non-Linear Layers

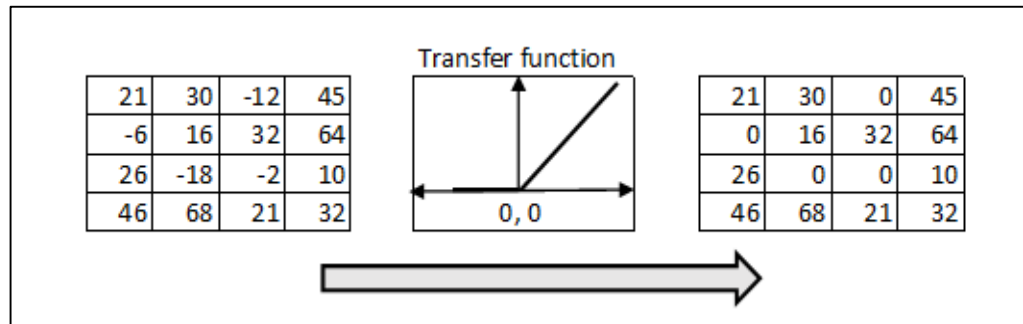
Typically neural networks and CNN depends or relies on a non-linear “activation” or “activation” function to fire distinct identification of likely features on each hidden layer. Convolutional neural networks might use a different type of specific functions, for example rectified linear units (ReLU) or non-linear functions such as hyperbolic tangent, sigmoid or softmax to implement non-linear triggering.

Rectified Linear Unit

A ReLU function is implemented by $\varphi(x) = \max(x, 0)$. Therefore input and output sizes of the layer are the same. Mainly, the Relu functions increases the nonlinear properties of the trigger function as well as of the overall network without affecting convolution layer’s receptive fields. The main advantage of a ReLU is that training time of the network is way faster comparatively to other

non-linear functions used in CNNs (e.g., hyperbolic tangent, softmax and sigmoid). ReLU functionality is presented in Figure 2.20 and transfer function is plotted between input and output matrices.

Figure 2.20: Representation of ReLU function.



Continuous trigger (non-linear) function

For the each feature, the non-linear layer process element by element. These non-linear functions can be hyperbolic tangent, sigmoid or softmax in order to process output with classification problems.

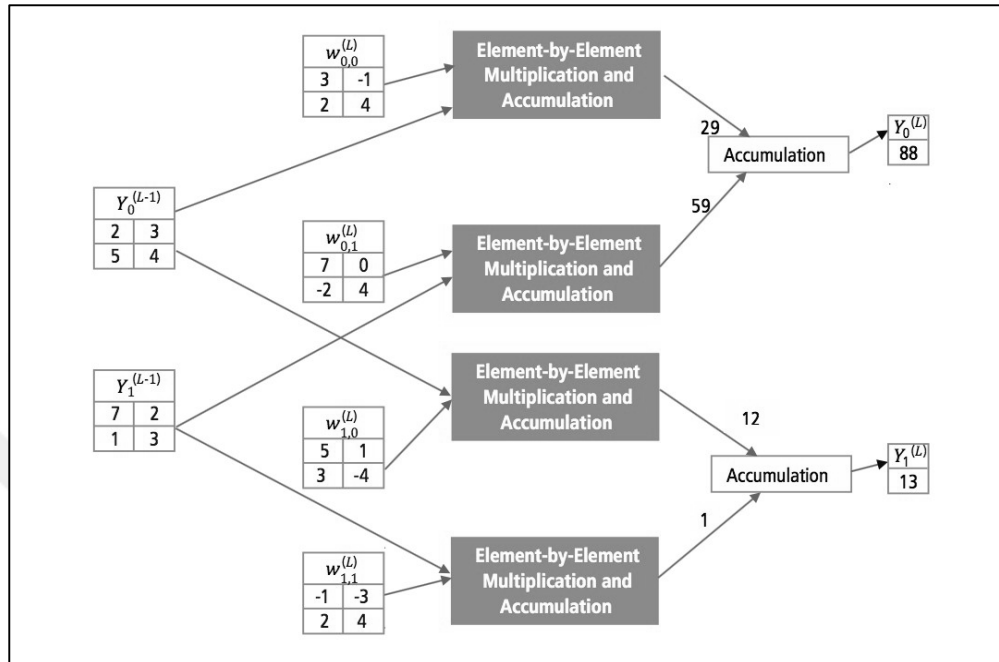
FC (Fully-Connected) layers

FC (Fully connected) layers are mainly used and constructed as the final layers of a CNN. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular artificial neural networks. Their activations mathematically sum a weighting of the features from the previous layer which can be done by matrix multiplications and its main aim is determining to target which can be accepted as classification. In these fully connected layer, all feature elements that fed by of the previous layer are used for calculating the output feature.

Figure 2.21 shows the FC (fully-connected) layer L where L-1 has got two features and each of composed of 2x2 matrices. Figure 2.22 shows stack of

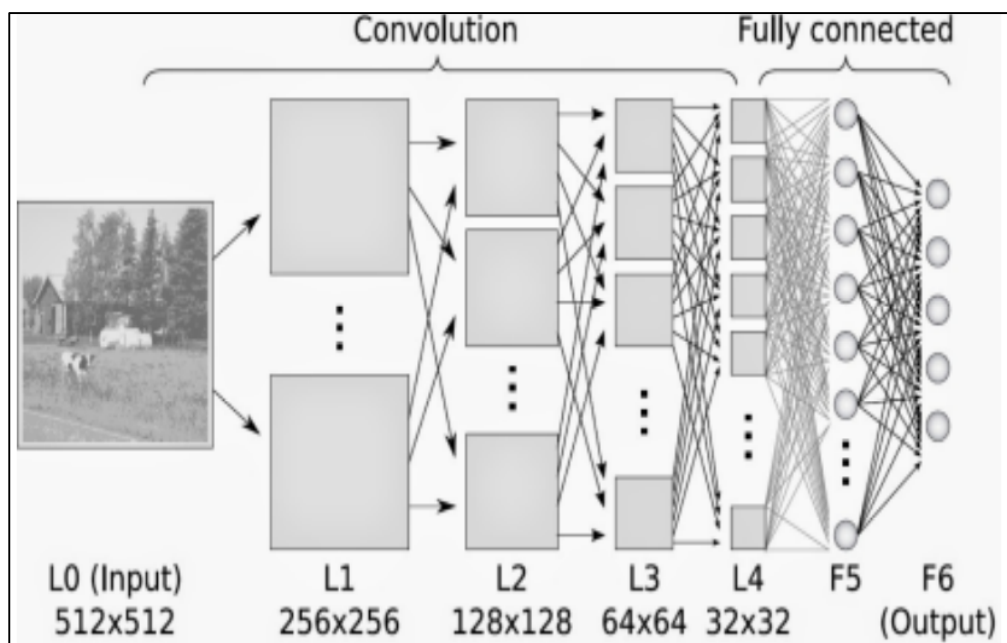
convolutional layers and fully connected layers to represent image classification process.

Figure 2.21: Fully connected layer processing



Source : Hijazi, S., Kumar, R., & Rowen, C., 2015. Using Convolutional Neural Networks for Image Recognition.

Figure 2.22: Convolution and fully connected layers

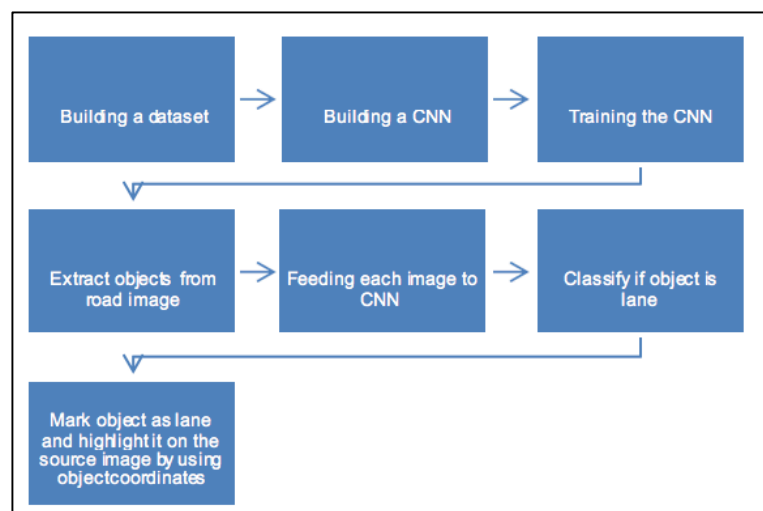


3. METHODOLOGY

In this chapter, implementing vehicle road lane detection system with convolutional neural network will be explained. All the implementation was done by Java platform. For building neural network deeplearning4j project is used. OpenCV java wrapper JavaCV is used for image processing and generating dataset. Dataset is downloaded from Caltech. In addition to Caltech dataset, manually videos are taken by driving in Istanbul city roads and many lane objects are extracted from the dataset. Test application is also build on Java Swing platform.

Building a proposed lane detection system based on convolutional neural networks, following steps were taken in thesis. Creating a dataset for train convolutional neural network, building a convolutional neural network, training CNN and measure performance with test set. Save network with weights and all relevant parameters as a image file, extract object proposals from given image and apply basic filter on them, give each object to neural network to classify if it is a lane object, if the object is classified as lane then high- light that object on the given source image. Figure 3.1 is presenting basic flow of the proposed method.

Figure 3.1: Flow of proposed method.



3.1 COLLECTING and BUILDING DATASET

In the field of machine learning, feeding ml algorithm with the training data as much as possible will increase the accuracy of the model. Typically to create a machine learning application, the first step is collecting dataset and training the algorithm to find optimal parameters of the model. Similarly, training dataset is collected to train CNN which will classify image objects and reports its class is lane or not. In order to create data set two process is followed. First of all, Caltech Cordova dataset is downloaded from the internet. In addition to adding more data to training set, manually videos are taken by mobile camera during the driving. Figure 3.2 represents the sample images from Caltech dataset and Figure 3.3 video image taken by camera.

Figure 3.2: A Sample image from Cal-Tech Cordova dataset.



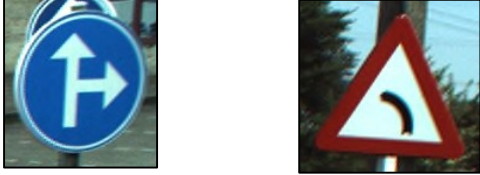



Figure 3.3: Sample road image from camera video.



Example image that is taken by camera is not a direct part of the training process. In the example image in Figure 3.3, the field of view contains bridge, lanes, vehicle on the road, skyscraper and some building and the highway road. Thus lane objects should be extracted from the given image and save under training set by labeling it. Therefore, one application is created and it extracts objects from the given images. After all those objects are extracted, lane objects are labeled with hand. Table 3.1 represents sample objects from the training dataset.

Table 3.1: Sample objects from training-set.

Lane object	
Tree objects	
Traffic signs	
Grass object	

All object images are coded by 24 bit RGB under PNG file format under file system of host computer and approximately size of 10 KB per each file.

As presented in Table 3.1, in addition to lane objects, some of irrelevant objects related to roads are added to training dataset such as tree objects, traffic signs and grass objects. The aim of adding irrelevant objects is to increase the accuracy of neural network. Other- wise it might classify some shapes in the traffic sign as lane since the characteristic of the shape similar to lane geometrical model. Adding such

irrelevant objects increases correlation of the samples and the accuracy. In the Figure 3.2, the table represents number of the objects in the training dataset.

Table 3.2: Numbers of training objects and labels.

Lane Objects	978
Tree	328
Traffic Signs	1110
Asphalt	299
Grass	70
Cordova irrelevant shapes	43
Irrelevant objects	112
Territory	7
Total	2947

3.2 ESTABLISHING and TRAINING CNN

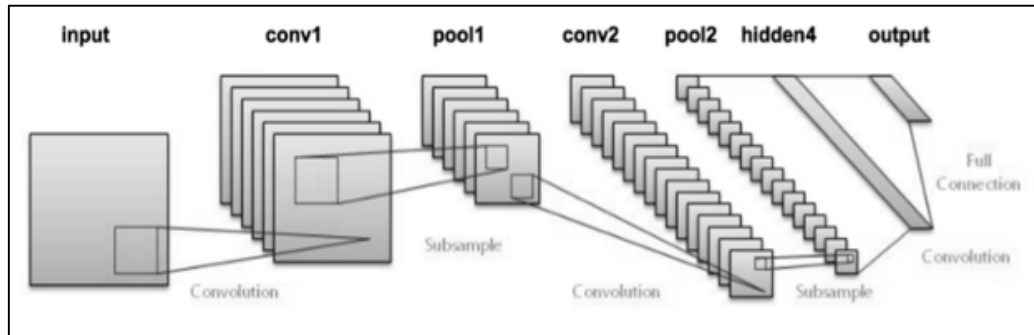
In order to build a CNN and utilize it for classifying objects that are proposed, a different multilayered CNN architectures were proposed by researchers. However, these architectures that are proposed mainly depends on computing resources where memory, storage and multicore computing advantages give researchers to implement deeper networks in a modern computer. Following architectures have been proposed by researchers.

Lenet is the first and produce good performance of convolutional neural networks that was developed by Yann LeCunn in (1990s). After LeNet application was proposed, many OCR application has been implemented by LeNet architecture,

reading digits and recognizing characters of handwritten texts. AlexNet is another popular network architecture where convolutional neural networks become popular. In (2012), Krizhevsky et al. submitted the architecture to the ImageNet ILSVRC (ImageNet Large Scale Visual Recognition Challenge) challenge and it significantly outperformed with top 5 error of 16%. The network was very close to LeNet architecture but it added many convolutional layers that deeper, bigger and featured. ZFNet is ILSVRC 2013 winner by Matthex Zeiler and Rob Fergus. It was an improvement on AlexNet by setting on some parameters such as expanding the size of middle convolutional layers and using smaller stride and kernel size on the first layer. Google proposed GoogLeNet which is the winner of ILSVRC 2014. The main contribution of it was the Inception Module which allows reducing the network's number of parameters. In addition to those developments, this network uses Average Pooling instead of Fully Connected layers at the top of the Convolutional network for eliminating a large of amount of parameters. VGGNet is the second of ILSVRC 2014, it was the network by Karen Simonyan and Andrew Zisserman. The main contribution was, the network depth which is a critical part for good performance. VGGNet architecture contains 16 Convolution and fully connected layers, from beginning to end of the network that it uses 3x3 kernel for both convolutions and pooling. However, downside of VGG- Net is that it is more expensive to evaluate and uses a lot of memory and parameters. ResNet is a residual network developed by Kaiming He and his friends. It was the winner of ILSVRC 2015 with its special feature of connection skipping and use of batch normalization which allows faster learning rate and higher accuracy in overall network. The Architecture is also missing fully connected layers at the end of the network.

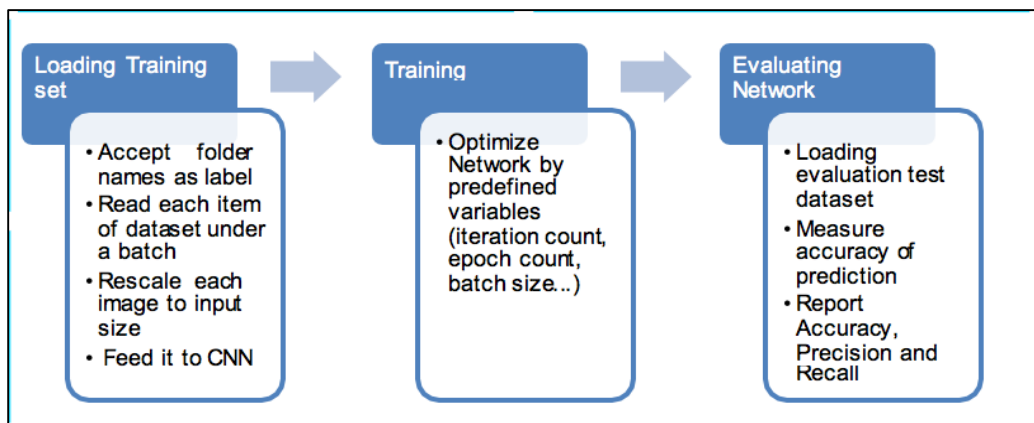
LeNet Architecture is selected as lane detections system by utilizing convolutional neural network. Following Figure 3.4 represents the layers of LeNet architecture.

Figure 3.4: Architecture of LeNet.



In order to implement a CNN, a Java based deep learning library called DL4J is used. DL4J (Deep learning for java) is an open-source, distributed deep-learning project in Java and Scala by the data science company SkyMind and aims at building deep learning applications for enterprise level. The project is composed of many deep learning algorithms, distributed computing support and many dataset utilities such as image I/O operations, speech and text reading helpers in order to vectorise them to present data. By using DL4J and collected dataset, following method that is presented in Figure 3.4 is taken for training our convolutional network to solve classification problem during lane detection process.

Figure 3.5: Training method flow.



Setting variables of network and training process is the key point of getting sufficient network classification as a result. Thus, following variables were set during training process and varied by output accuracy.

Input Size is set 64x64 pixel where each image of training and test set rescaled to this dimension.

Channel is set to 3 where each image is coded under RGB format. Each channel is mapped to color channel of image.

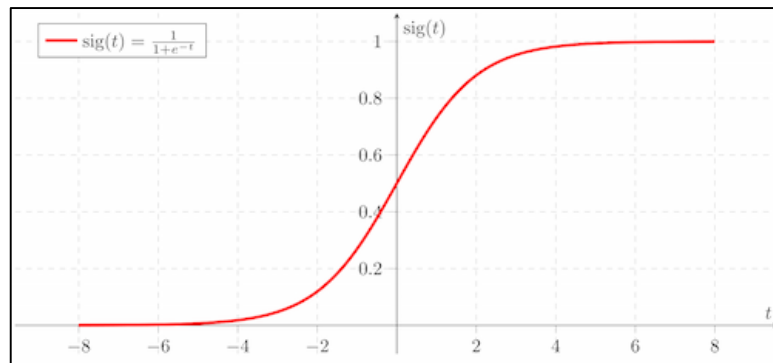
Batch size is the number of examples to be fetched with each step.

Epoch is complete pass through a given dataset.

Number of labels is output count of the network.

Weight Initialization is important parameter where adjusting hyper parameters during experiment process. Generating weight parameters for each experiment may vary on accuracy because initial weights can lead algorithms to different local minima and errors- cape. Also the choice of weights is important. When weights in the network are too small, then the signal shrinks as it passes through each layer where it become too tiny to be useful. When the weights in the network are too large, then the signal grows while it passes through each layer until it's too big to be useful. Suppose we are using sigmoid function as activation function. Sigmoid function is approximately linear when input goes close to zero, if the weights are too large, sigmoid function become flat for larger values as it is shown in figure 3.6.

Figure 3.6: Plot of sigmoid function.



Due to this fact, initializing the network weights properly is very important to make network function properly. During implementation of LeNet architecture in this thesis, Xavier weight initialization(Xavier et al. 2010) method is used which initializes weights in the network by drawing them from distribution with zero mean and specific variance in other words, it keeps the variance remain the same while it passes each layer,

$$\text{Var}(W) = \frac{2}{n_{in} + n_{out}} \quad (3.1)$$

Where W is the initialization for the neuron, n_{in} is the number of input neurons and n_{out} is the number of output neurons.

Optimization Algorithm is the method of optimizing cost function. Stochastic Gradient Descent is utilized to help minimize error.

Iteration Count is a learning step for updating model's weights. The network is fed by the data, makes predictions about the data, and then corrects its own parameters based on error which shows how wrong its predictions were. More iterations allow network to take more steps and learn more which means minimizing error.

Learning rate is the step size of optimization algorithm where it adjusts weights with each iteration. A high learning rate allows net traverse quickly but very error-prone however low learning rate is more likely to find the minimum but it will do this very slowly.

Momentum is the additional factor in determining how fast an optimization algorithm converges on the optimum point.

Regularization is the technique to prevent overfitting. Overfitting is when the model fits the training data very well but performs poorly when network predicts the output of data which does not belong the dataset.

After variables and parameters are set, network is fit by training dataset and start learning process. After the process is done which depends on the epochs and iterations, network evaluates its performance by testing predictions with test dataset. Measurement parameters of the network,

Accuracy is the percentage of test images that were correctly identified by the network.

Precision is the number of true positives divided by the number of true positives and false positives.

Recall is the number of true positives divided by the number of true positives and the number of false negatives.

F1 Score is weighted average of precision and recall.

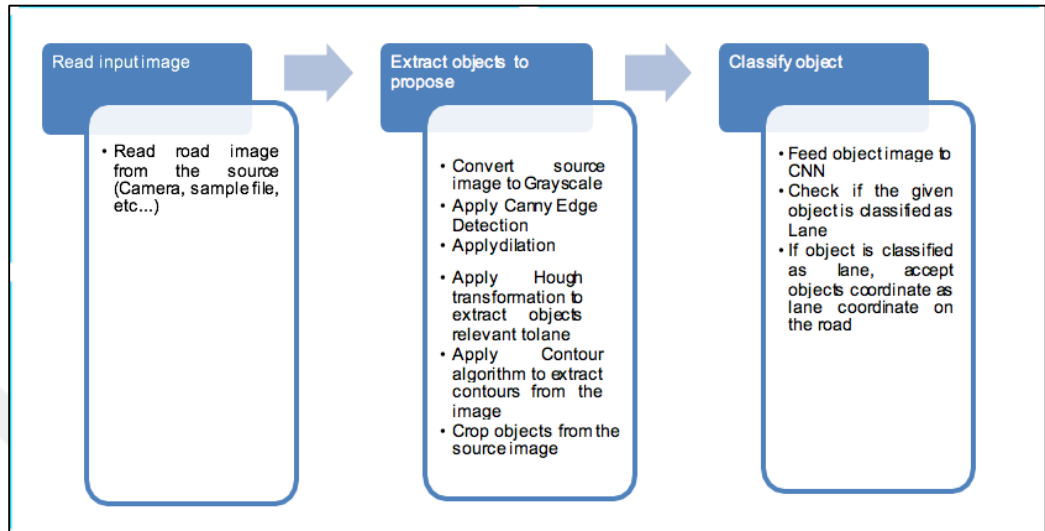
After the network is trained, the network is saved as an image file under a filesystem when it is desired to be used for image classification. In this thesis, network image is compressed and saved as zip file in order to be kept small.

3.3 PROCESSING ROAD IMAGE

In order to detect lines on the road, first the objects in the source image should be processed to extract objects. After this process is done, all object that are extracted is given to CNN in order to classify it whether it is recognizable by network. If the network classifies the given object as “Lane”, then our system becomes able to

access coordinates of the lane on the road. Figure 3.6 represents the flow of image processing and classifying the object by the network.

Figure 3.7: Flow of image processing and classification.



Implementing image processing and related computer vision algorithms, a java wrapper of OpenCV is used. OpenCV is most common open-source computer vision library that is being used for developing computer vision applications.

Reading image from the source

An example image from the camera or the Caltech Cordova dataset is used for our testing. Figure 3.7 shows the samples from the source images.

Figure 3.8: Samples from source road images.



Suppose I is the original image. After the source image I is loaded into memory, it is converted to Greyscale format in order to apply canny edge detection. Transformation to Grayscale by $Y \leftarrow 0.299.R + 0.587.G + 0.114.B$. Figure 3.8 shows the source image in 24 bit RGB color format in left section and 8 bit Grayscale format in right section,

Figure 3.9: Greyscale color transformation on sample image.



Applying grayscale transformation of the image will allow edge detection to perform well on the image. Edge detection application is used for extracting objects especially lanes on the road from the source image.

OpenCV Canny edge detection algorithm is employed for extracting lanes as well as other objects from the given source image. Regarding to edge detection implementation in OpenCV, Gaussian filter is used firstly for filtering out any noise on the source image. Since all edge detection results are easily affected by image noise, it is important to filter out the noise preventing false detection caused by the noise which means it is for smoothing the image by convolve kernel with the given image. Kernel of *size* = 5 shown in below matrix is used since that size is common and useful for many images,

$$K = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (3.1)$$

After filtering out of the noise on the image, the algorithm finds the intensity gradient of the image by using Sobel filters. It applies max of convolution pair in x and y directions,

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad (3.3)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (3.4)$$

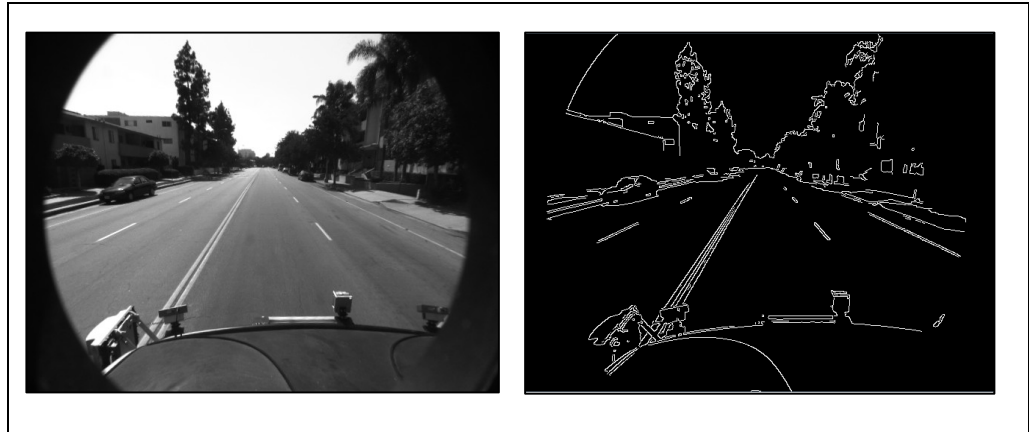
And it finds the gradient strength and direction with,

$$G = \sqrt{G_x^2 + G_y^2} \quad (3.4)$$

$$\theta = \arctan \left(\frac{G_y}{G_x} \right) \quad (3.5)$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0° , 45° , 90° and 135° for example). After the gradient strength and direction is found, it applies *Non-Maximum* suppression where it removes pixels that are not considered to be part of an edge. As a final step, *Hysteresis* step is taken. Canny does use two threshold. One is upper threshold where the pixel is accepted as an edge if the pixel gradient value is higher than the upper threshold or it's rejected if the pixel gradient value is below the lower threshold. Canny recommended an upper: lower ratio between 2:1 and 3:1. Below images that shown in Figure 3.9 presents the given source image and processed image after canny edge detection is applied.

Figure 3.10: Input image, Edge detected image.



Since canny edge detection operation outputs a binary image of the road and extracts the lanes and other objects, in order to make it strong or fill up empty pixels in the lanes, a dilation morphological operation is applied. Suppose f is binary input image and convolved with 3×3 kernel k . Let,

$$c = f * k \quad (3.6)$$

And Dilation is

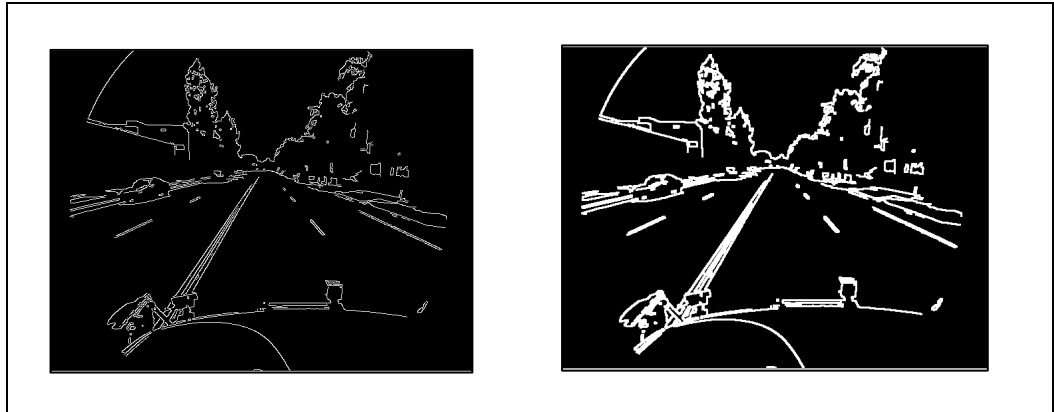
$$dilate(f, k) = \theta(c, 1) \quad (3.7)$$

Where

$$\theta(f, t) = \begin{cases} 1, & f \geq t \\ 0, & else \end{cases} \quad (3.8)$$

In the figure 3.10, the output shows the source binary image and output binary image that dilation is applied.

Figure 3.11: Input image, processed image with dilation.



Dilation process made the edges stronger in the binary image before the next method where the lines in the image will be found. This operation is called Hough Transformation.

The Hough transform is a feature extraction method that is used in computer vision and image processing applications. The purpose of the method is to find certain class of shapes in the given binary image such as lines or circles. In 1962 Paul Hough described and patented the transform (Hough, 1962). It was a point to curve transformation which has many applications in pattern recognition. Later on, Richard Duda and Peter Hart (1972) invented the universal usage by calling “Generalized Hough Transformation” by applying Hough line transformation which is mainly used for finding lines in the image, suppose the straight line,

$$y = mx + b \tag{3.9}$$

Can be represented as a point (b, m) in a parameter space. However, due to characteristic of the vertical line, there can be infinite value of slope parameter m . Since computers do have finite resources, it could be computationally impossible to store slope parameter. Thus, Hough line transform proposes the use of polar coordinate system to represent the line parameters with (p, θ) where p is the distance from the origin and θ is the angle between the x axis. Therefore,

it is possible to associate with each line of the pair (p, θ) . For Hough Transform the line is expressed in polar coordinate system as the equation,

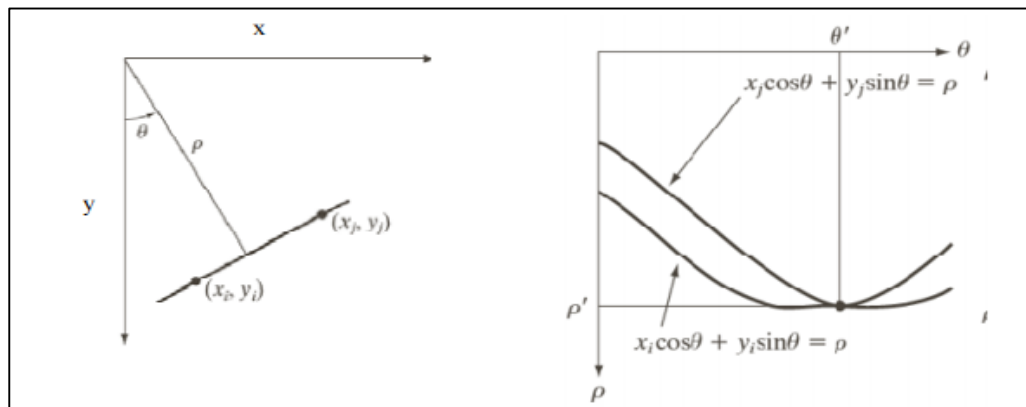
$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{p}{\sin\theta}\right) \quad (3.10)$$

Also

$$p_\theta = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta \quad (3.11)$$

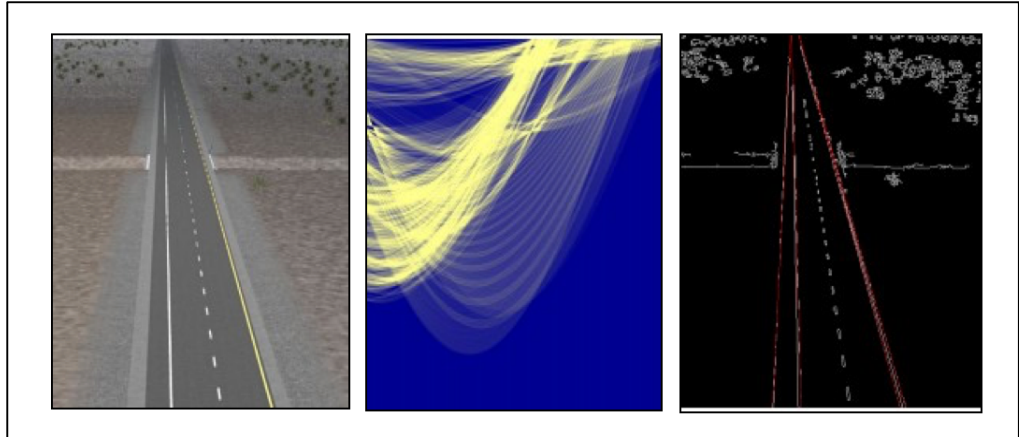
Which means each pair (p_θ, θ) represents each line that passes by (x_0, y_0) . According to algorithm that by calculating each line passes by the given non background pixel (pixel value is not 0) and storing the parameters in the accumulator table, it will allow to find lines. In the Figure 3.10, suppose there is a line starts from (x_i, y_i) to (x_j, y_j) . When the each pixel on the given line is scanned by Hough transformation algorithm, and the each pair (p_θ, θ) is stored in the accumulator table, as it seen on the intersection point (p', θ) corresponds tot he lines that passes through given (x_i, y_i) to (x_j, y_j) .

Figure 3.12: Hough transformation visualization.



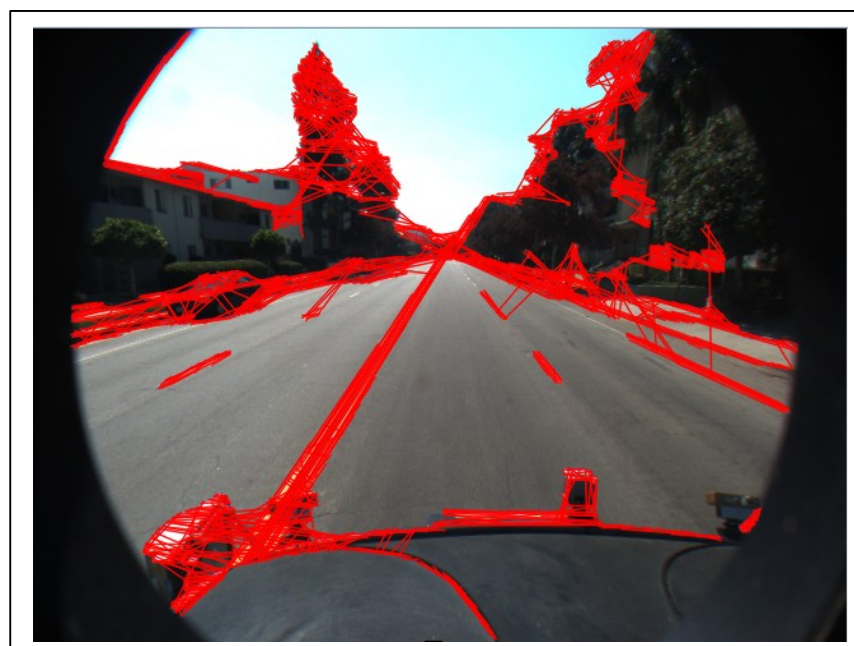
Source: <http://www.uio.no/studier/emner/matnat/ifi/INF4300/h09/undervisningsmateriale/hough09.pdf>

Figure 3.13: Hough space and lanes on the source image. (Left: original image, Middle : Hough Space, Right : Lines found on the image)



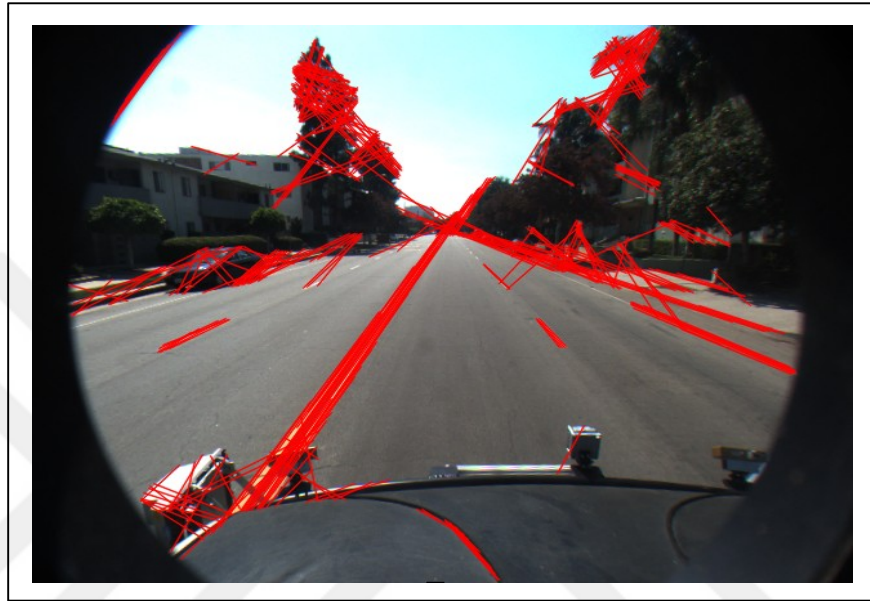
Using Hough transformation to find lanes on the given road image, transformation will report the lines on the image by using OpenCV Hough transformation. In the figure 3.12, the lines that are caught on the image is shown. The red lines as it is seen on the images are the lines found by Hough transformation and they are drawn on the image.

Figure 3.14: Lines are found on the given image by Hough line transformation.



Even though, the lines are found on the road image, in order to eliminate irrelevant shapes, a simple filter is used for eliminating unnecessary lines. The filter does basically eliminates lines whose slope is larger than 30° and area of the line smaller than 400px. Figure 3.14 shows the output after elimination,

Figure 3.15: Filtering out lines on the given image.



3.4 OUTPUT

Convolutional neural network typically used as classification in image recognition process. However, as it's mention, the lane detection is interested in lanes on the road instead of the road image given. Therefore a method should be involved at this phase to extract relevant object which might contains lane. This method is called Object proposing. Therefore, the lines that is found by hough transformation is used to extract object from given image. Lets say, the line is found from the coordinates (x_1, y_1) and (x_2, y_2) .). Cropping the rectangle by the given coordinates will create a new image on the memory which only contains the object that the line is passing over. This subimage will be considered as an object for being proposed to CNN to classify if it is a lane object or not. By traversing on all lines and extracting each subregion of the

image by CNN will help to find lanes on the road. In the figure 3.15, the left image is the input image and the right image contains the regions of the lanes after CNN classified each object.

Figure 3.16: Processed final image. Left: Source image, Right: Output image marked with lane objects



4. RESULTS

Lane detection system recognition accuracy depends on two different method on the system proposed. One is computer vision algorithm to extract objects from the road data to find objects similar to lane shape and CNN classification method which detects if the proposed object is lane or not. Since, CNN become a lane identification component of the system, it's very important to adjust CNN parameters and train it properly to reach acceptable result. In order to approach acceptable result of CNN, some experiments with different parameters will be done. These parameters might be training set data count, epoch, training data image sizes and network internal parameters such as learning rate, regularization, layer count etc. Since LeNet architecture does have own learning rate, regularization and layer count, the other parameters will be considered. In order to observing CNN classification accuracy, different parameters will be set and training phase will be re-run. The parameters and their values before each test is shown in Table 4.1

Table 4.1: CNN Training Parameters.

Parameter	Values	Values
Input Image Size	32x32 Pixels	64x64 Pixels
Epoch/Iteration Count	50, 200, 800, 1600	50, 200, 800, 1600
DataSet Input Image Count	2947, 2943	2947, 2943
Dataset Training Set Image Count	46, 486	46, 486
Dataset Label Count	28	28
Dataset Label Count (Lane/NoLane)	2	2

Each parameter-set (image size = 32x32 pixels, Epoch = 50), the training application run by those parameters to train model. After training by using test-set, evaluation of the model is saved. There is also two type of training method is tried during training phase. One is splitting training dataset into 28 different label (Lane,

tree, grass, traffic sign, stops sign etc....) and another one is two different label which is Lane or not a lane for testing and observing classification process accuracy according to these two different approach. There is also, different test dataset size is used where in one case (28 different label) contains 46 different test objects, 489 different objects are used for testing 2 labeled class method.

The parameters and evaluation results of each parameter-set for 28 different class is shown in table 4.2

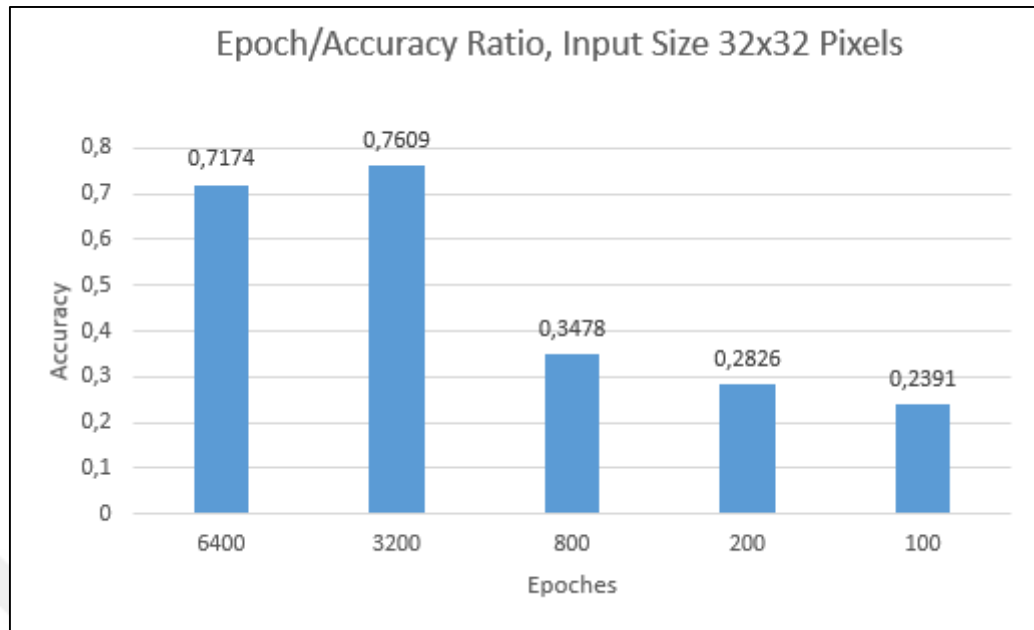
Table 4.2: Evaluation and result of each parameter-set (28 classes).

Input Image Size	Epoch/Iteration Count	Accuracy	Precision	Recall	F1 Score
64x64	1600	0,8478	0,9467	0,8913	0,9182
64x64	800	0,8043	0,902	0,8333	0,8663
64x64	400	0,5870	0,7175	0,6111	0,6601
64x64	100	0,3478	0,4520	0,2391	0,3128
64x64	50	0,2826	0,45	0,1594	0,2354
32x32	6400	0,7174	0,8581	0,8285	0,8430
32x32	3200	0,7609	0,8465	0,7923	0,8185
32x32	800	0,3478	0,5327	0,2899	0,3474
32x32	200	0,2826	0,4324	0,1594	0,2329
32x32	100	0,2391	0,2894	0,1014	0,1502

As it is seen in the table 4.2, increasing epoch on learning will result better accuracy since more iterations allow network to take more steps and learn more which means minimizing error. However, for the pixel size of 32x32 where it is input image size, in the table 4.2 it is seen that Epoch 6400 results lower accuracy than epoch 3200. It is caused by Over-Fitting which means the weights and parameters are fit on training set but results improper predictions when the source does not belong to training set. It is also important that the input image size is important for learning accuracy as it is seen in the table 4.2. However, as it is seen in the both table 4.2 and figure 4.1 and Figure 4.2, 64x64 pixel size of training objects give better performance than 32x32 pixels size.

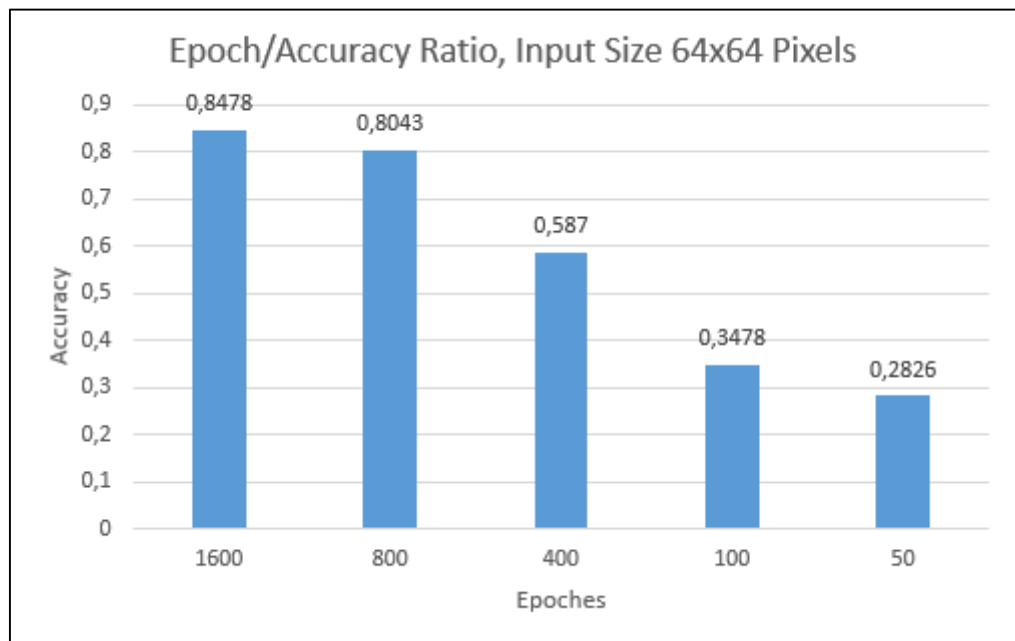
As it is seen in the Figure 4.1, even though epoch is increased after 3200 iteration, the accuracy does not increase even it is accuracy goes down as it is mentioned as over-fitting.

Figure 4.1: Epoch/Accuracy ratio of 32x32 pixel size.



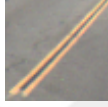





However, when training image size is selected to 64x64 pixels, as it is seen in the table 4.2, with the same iteration count, it is reaching better accuracy than 32x32 pixels training images.

Figure 4.2: Epoch/Accuracy ratio of 64x64 pixel size.



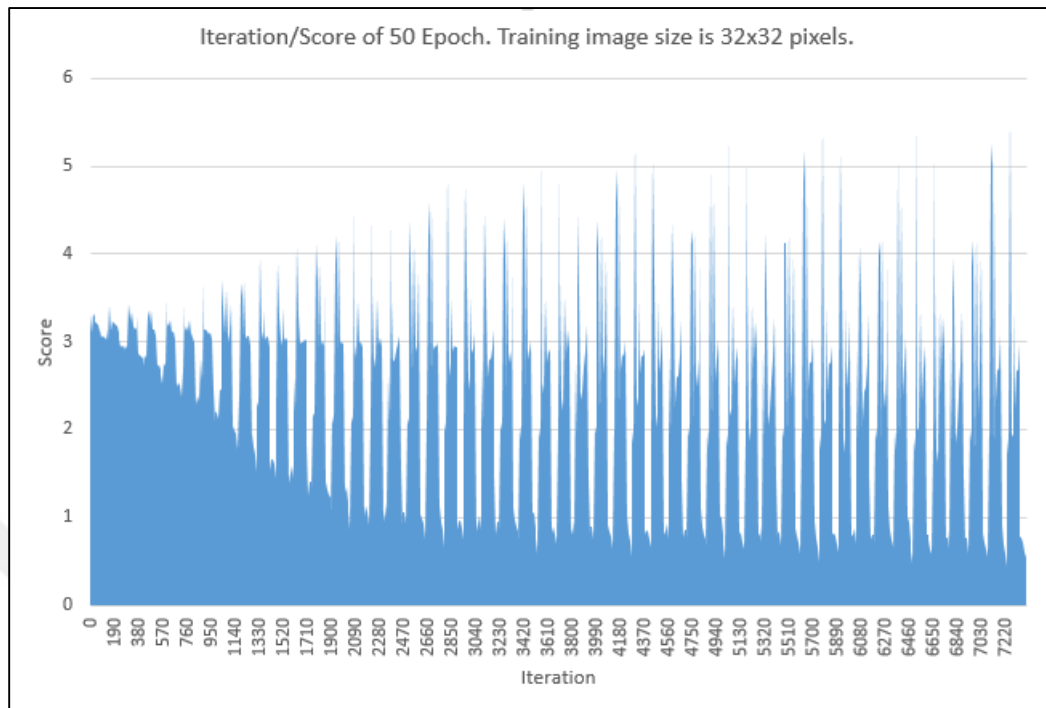
In order to understand the role of training image size relation to accuracy, let's check the sample images in Table 4.3. As it is shown, 64x64 resized form of given training image provides better information to CNN to learn edges, corners and other relevant shapes. Therefore to using better resolution for training image increasing the accuracy. However, it will cause learning phase to consume more time and computational resources since input data is bigger.

Table 4.3: Training images resized versions.

Original Image	32x32 pixels resized	64x64 pixels resized
		
		

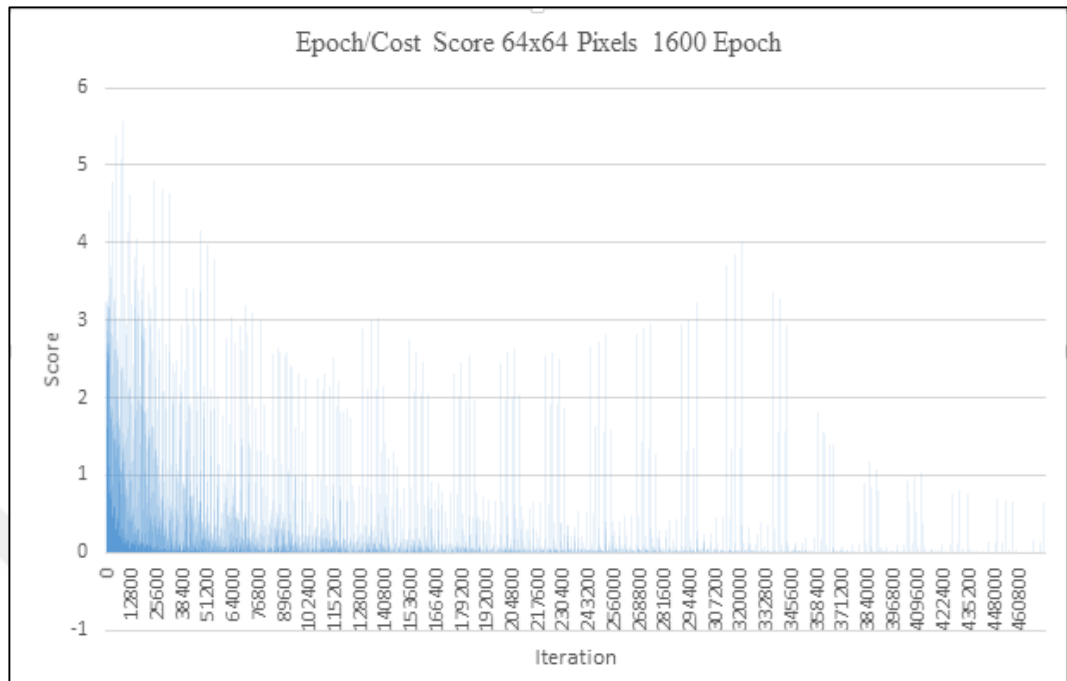
Regarding to accuracy, it is important to check score function steps during training. In our proposed solution, the each 10 step is logged for creating graph of score function divergence. Let's look at Figure 4.3 showing the score function where vertical axis shows score result of cost function and horizontal axis is iteration count.

Figure 4.3: Score function optimization of 32x32 pixel size.



The plot of score functions, it is seen that there is many peak points throughout the iterations. It is caused by mini-batch where SGD processes part of training data for each time. However, as it is seen, SGD approaches to minimum until training is finished on specified epoch (It this sample it's stopped on iteration count on 7220). However, when input size is set to 64x64 pixel size and epoch is set to 1600, as it is seen in the Figure 4.4, Cost function approaches to almost 0 when epoch is increased. Therefore, it proves that generally more epochs and better image sizes resulting better accuracy for Convolutional neural network applications. Regarding to over-fitting, the learning phase could be stopped when the cost function score approaches to almost 0 and keep this output stability for a specified batch count.

Figure 4.4: Score function optimization of 64x64 pixel size.



In addition to 28 different classes, the training dataset is splitted in to 2 different classes. One class is determining if the proposed object is lane, another class is determining if the proposed object is not a lane. According to test results of 28 different classes, it results that 64x64 pixels size of training dataset objects is produces more accurate result. Thus, 64x64 pixels size of training dataset is used for 2 classes test. Table 4.3 shows the results of training and test results.

Table 4.4: Evaluation and result of each parameter-set (2 classes).

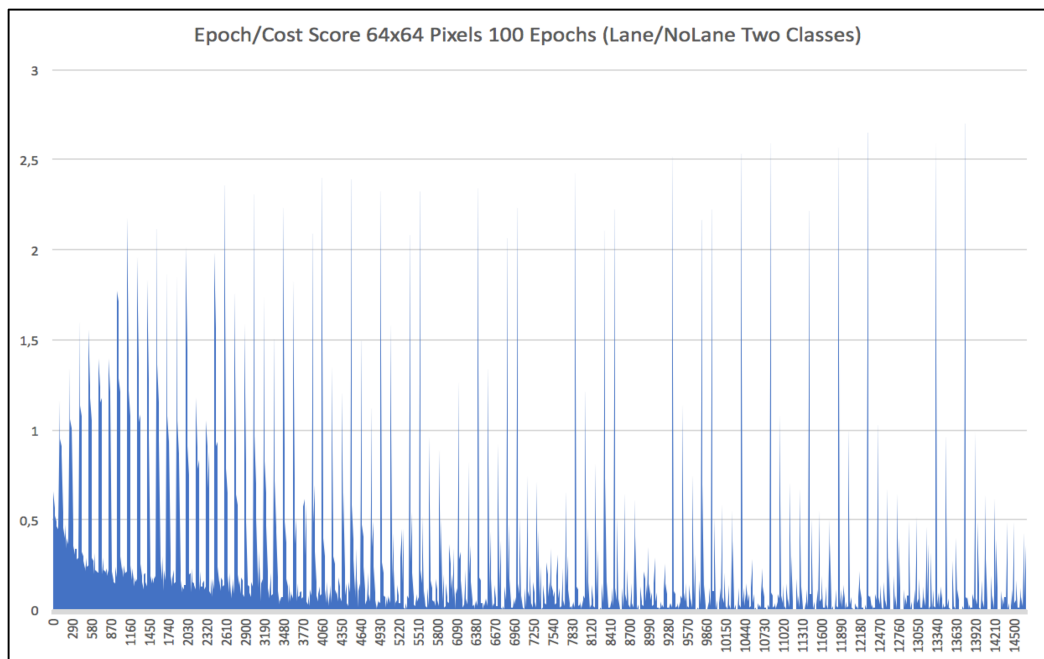
Input Image Size	Epoch/Iteration Count	Accuracy	Precision	Recall	F1 Score
64x64	1600	0,9506	0,9502	0,9554	0,9528
64x64	800	0,9403	0,9411	0,9461	0,9436
64x64	400	0,9444	0,9442	0,9494	0,9468
64x64	100	0,9527	0,9513	0,9559	0,9536
64x64	50	0,965	0,9635	0,9671	0,9653

Table 4.5: Output of predictions (2 classes).

Input Image Size	Epoch/Iteration Count	Number of NoLanes classified as NoLane	Number of NoLanes classified as Lane	Number of Lanes classified as NoLane	Number of Lanes classified as Lane
64x64	1600	245	24	0	217
64x64	800	240	29	0	217
64x64	400	243	26	1	216
64x64	100	249	20	3	214
64x64	50	255	14	3	214

It is seen in the table 4.3 and Table 4.4, it produced better performance to split classes in to two different classes which performs prediction if the given object is a lane or not a lane. There is also one important point that splitting dataset to two different classes approached acceptable accuracy by 50 epochs where the other method which contains 28 different classes is not able to reach the closed accuracy by 1600 epochs. It also shows that eliminating unnecessary classes from training dataset produces better accuracy for convolutional neural networks. Figure 4.5 shows score functions of two classes and 100 epoch optimization graph.

Figure 4.5: Score function optimization of 64x64 pixel size for two classes.



5. CONCLUSION

Many of the computer vision applications do extract and recognize objects from the image by using hand designed learning system on specific problem where those learning system is defined by mathematical parameters such as extracting parallel splines and lines from image to detect road lanes or finding shapes similar to face structure on the given image to detect faces on the image. These hand-designed mathematical models are strictly depend on the specific objects. For example, by using Hough line transformation to detect lanes on the road image, the implementation should extract lines, detect if the line objects are overlapped on vanishing point to understand if there is a perspective effect, if the line does not have any parallel line then the implementation should decide if it is accepted as lane or not, and each time to make the algorithm stronger, new road images should be reviewed by researchers and the algorithm should be updated. In such computer vision applications, convolutional neural network implementations supports those algorithms to detect or classify the sample objects to understand if it is expected object. Since convolutional neural network classifies the proposed object, the algorithm that extract and analyze objects can be simple where analyzing process become simpler. The proposed lane detection implementation, the object extracting phase is only extracts the line objects and implementation does not need any complex object analyzing phase. Convolutional neural network decides if the object is lane or not. Therefore, convolutional neural network supports increasing accuracy of such object classification solutions while it does not need any complex hand-designed mathematical parameters involved. In addition to ease of implementation, convolutional neural networks does help solutions to detect many different objects from the same image. For example, in the proposed lane detection method, by training CNN on vehicle, truck, motorbike objects, the same solution will classify vehicles on the road by only adding simple vehicle object extraction method. It is also same for detecting traffic lights on the same image. Therefore, convolutional neural networks will become common method in many computer vision object classification solutions where it supports algorithms to detect any object by allows implementation to eliminate complex operations.

REFERENCES

Books

- D. O. Hebb, 1949. *The organization of behavior: A neuropsychological theory*. New York, Wiley.
- T. M. Mitchell, 1997. *Machine Learning, 1st ed.* New York, NY, USA: McGraw-Hill, Inc.
- E. Alpaydin, 2010. *Introduction to Machine Learning, 2nd ed.* The MIT Press.
- Richard Szeliski, 2010. *Computer Vision: Algorithms and Applications*. London, Springer.
- I. Goodfellow & Y. Bengio, A. Courville, 2016. *Deep Learning*. The MIT Press.



Periodicals

- McCulloch, W. S., & Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115-133.
- Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.
- VC, H. P., 1962. *U.S. Patent No. 3,069,654*. Washington, DC: U.S. Patent and Trademark Office.
- Duda, R. O., & Hart, P. E., 1972. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), 11-15.
- Werbos, P. J., 1974. Beyond regression: new tools for prediction and analysis in the behavioral science. *Ph. D. Thesis, Harvard University*.
- Canny, J., 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679-698.
- Jochem, T., Pomerleau, D., Kumar, B., Armstrong, J., 1995, PANS: A Portable Navigation Platform, *IEEE Symposium on Intelligent vehicle*, September 25-26, 1995, Detroit, Michigan, USA
- Chen, Mei, Todd Jochem, and Dean Pomerleau. 1995, AURORA: A vision-based roadway departure warning system. *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*. Vol. 1. IEEE.
- D. Pomerleau and T. Jochem, 1996, Rapidly adapting machine vision for automated vehicle steering, *in IEEE Expert*, vol. 11, no. 2, pp. 19-27, Apr 1996.
- B. M. Broggi, 1998. GOLD: A parallel real-time stereo Vision system for generic obstacle and lane detection, *IEEE Transactions on Image Processing*, pp. 4-6.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. , 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- C. Kreucher and S. K. Lakshmanan, 1998. A Driver warning System based on the LOIS Lane detection Algorithm, *Proceeding of IEEE International Conference On Intelligent Vehicles*. pp. 17 -22.
- E. D. Dickmanns, 2002, "The development of machine vision for road vehicles in the last decade," *Intelligent Vehicle Symposium, 2002. IEEE*, 2002, pp. 268-281 vol.1.
- Y. Wang, E. K. Teoha, D. Shen., 2004. Lane detection and tracking using B-Snake, *In: Image and Vision Computing* 22, pp: 269-28.

- Jung, Cláudio Rosito, and Christian Roberto Kelber, 2005. Lane following and lane departure using a linear-parabolic model. *Image and Vision Computing* 23.13 : 1192-1202.
- Sun, T. Y., Tsai, S. J., & Chan, V., 2006. HSI color model based lane-marking detection. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE* (pp. 1168-1172). IEEE.
- Aly, M., 2008. Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium, 2008 IEEE* (pp. 7-12). IEEE.
- Kim, Z., 2008. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1), 16-26.
- Khalifa, O. O., Hashim, A. H. A., & Assidiq, A. A., 2009. Vision-based lane detection for autonomous artificial intelligent vehicles. In *Semantic Computing, 2009. ICSC'09. IEEE International Conference on* (pp. 636-641). IEEE.
- Q. Lin, Y. Han and H. Hahn, 2010. Real-time lane departure detection based on extended edge-linking algorithm, In *IEEE 2nd International Conference on Computer Research and Development*, pp. 725-730.
- Glorot, X., & Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats* (Vol. 9, pp. 249-256).
- Chang, C.C. and Lin, C. J., 2011, A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2(3), pp. 1 – 27
- Borkar, M. Hayes, M.T. Smith and S. Pankanti , 2011, A Layered Approach To Robust Lane Detection At Night , In *IEEE International Conference and Exposition on Electrical and Power Engineering, Iasi, Romania*, pp. 735 - 739, 2011.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
- Măriuț, F., Foșalău, C., & Petrisor, D., 2012. Lane mark detection using Hough transform. In *Electrical and Power Engineering (EPE), 2012 International Conference and Exposition on* (pp. 871-875). IEEE.
- Kim, K. B., Song, D. H., & Cho, J. H. (2012). Lane detection using fuzzy c-means clustering. *International Journal of Multimedia and Ubiquitous Engineering*, 7(4), 119-124.
- Simonyan, K., & Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

- Mistry, V.H.; Makwana, R. Survey, 2014. Vision based road detection techniques. *Int. J. Comput. Sci. Inf. Technol.* 5, 4741–4747.
- Zeiler, M. D., & Fergus, R., 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer International Publishing.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A., 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).
- Hijazi, S., Kumar, R., & Rowen, C., 2015. Using Convolutional Neural Networks for Image Recognition.
- Pallavi V. Ingale, Prof. K. S. Bhagat, 2016. Comparative Study of Lane Detection Techniques, *ICSTSD 2016 Track, International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC)*, ISSN: 2321-8169, PP: 381 – 390
- He, K., Zhang, X., Ren, S., & Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

Others Sources

- Turkish Statistical Institute, 2015, *Number Of Traffic Accidents And Results*, [Internet] http://www.turkstat.gov.tr/PreIstatistikTablo.do?istab_id=362 [accessed date 01 February 2017]
- National Highway Traffic Safety Administration, 2005, *Lane Departure Warning systems*, [Internet] https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/barickman_lanede_partuerwarning_final.pdf [accessed date 01 February 2017]
- Stanford University, U.S.A, Prof. Eric Roberts's personal web page, *Neural Networks*, [Internet] <https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Biology/index.html> [accessed date 19.03.2017]
- Wikipedia, *List Of Animals by number of neurons*, [Internet] https://en.wikipedia.org/wiki/List_of_animals_by_number_of_neurons [accessed date 19.03.2017]
- Stanford University, U.S.A, , Class of CS231 *Convolutional Neural Networks for Visual Recognition notes*, [Internet] <https://cs231n.github.io/> [accessed date 19.01.2017]
- Mohamed Alaa El-Dien Aly, *Caltech Lane Dataset* [Internet] <http://www.mohamedaly.info/datasets/caltech-lanes> [accessed date 10.06.2016]
- Deeplearning for Java, *An opensource deep learning framework for Java*, [Internet] <https://deeplearning4j.org/>. [accessed date 01.10.2016]
- OpenCV, *An opensource computer vision software library*, [Internet] <http://www.opencv.org/>. [accessed date 01.10.2016]