CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Mechanical Engineering

Department of Instrumentation and Control Engineering

Car assistant system for road lanes and traffic signs detection

2020

Submitted by: Shreetal Upadhyay

Supervisor: Ing. Jaroslav Busek, Ph.D.

Declaration of authorship

I hereby declare that this master’s thesis has been written by me in person. All information derived from other works has been acknowledged in the text and the list of references.

In Prague: 19. 08. 2019

Shreetal Upadhyay

Abstract

This master thesis is based on the car assistant system for recognizing road lanes and German traffic signs. The goal of the thesis is to design CNN architectures for classifying road symbols using several libraries like OpenCV. Keras and Tensorflow. The image classification analysis is based on the German Traffic Sign Recognition Benchmark using LeNet model under different Parameters such as Convolutional layer, flatten, Dropout ,Depth size,. Lanes detection Parameters are discussed using hough Transform , at the end udacity self driving car engine is simulated by collecting data from manual mode then trained to drive itself in autonomous mode using different techniques.

.

Keywords: Convolutional Neural Network, Traffic sign Classification, German Traffic Sign Recognition Benchmark, Lane detection ,Udacity Car engine.

Abstraktní

Tato diplomová práce je založena na asistenčním systému pro rozpoznávání silnic a německých dopravních značek. Cílem práce je navrhnout architektury CNN pro klasifikaci silničních symbolů pomocí několika knihoven, jako je OpenCV. Keras a Tensorflow. Analýza klasifikace obrázků je založena na německém benchmarku pro rozpoznávání dopravních značek pomocí modelu LeNet pod různými parametry, jako je vrstva převratu, zploštění, výpadek, velikost hloubky. Detekce jízdních pruhů Parametry jsou diskutovány pomocí transformace, na konci je simulován samotný motor automobilu s řízením sběru dat z manuálního režimu a poté vyškolen k tomu, aby se sám řídil v autonomním režimu pomocí různých technik.

Klíčová slova : Konvoluční neuronová síť, Klasifikace dopravních značek, Benchmark německého dopravního značení, Detekce jízdních pruhů, Motor automobilu Udaci

Acknowledgements

I would like to express my thanks to my supervisor Ing. Jaroslav Busek, Ph.D., for his valuable time, patience and guidance. I would also like to thank my parents for their immense support during the time of my graduation, a sincere thanks to my Friend Mr. Varun Burde for his timely help. Lastly I want to thank all my friends who helped me during my Graduation.

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Upadhyay Shreetal**	Personal ID number:	**473176**
Faculty / Institute:	**Faculty of Mechanical Engineering**		
Department / Institute:	**Department of Instrumentation and Control Engineering**		
Study program:	**Mechanical Engineering**		
Branch of study:	**Instrumentation and Control Engineering**		

II. Master's thesis details

Master's thesis title in English:

Car assistant system for road lanes and traffic signs detection

Master's thesis title in Czech:

Asistenční systém pro detekci dopravních pruhů a dopravních značek

Guidelines:

1. literature search on a given topic regarding neural application deployment
2. design a neural network for road lanes and traffic signs recognition
3. provide appropriate classification of road lanes and traffic signs
4. implement real-time assistant system for driver on a simulated HUD of a car
5. collect data from udacity self driving car engine simulator from training mode and validate the system function

Bibliography / sources:

ALY, Mohamed. Real time detection of lane markers in urban streets. In: 2008 IEEE Intelligent Vehicles Symposium. IEEE, 2008. p. 7-12.
FANG, Chiung-Yao; CHEN, Sei-Wang; FUH, Chiou-Shann. Road-sign detection and tracking. IEEE transactions on vehicular technology, 2003, 52.5: 1329-1341.
ARCOS-GARCÍA, Álvaro; ALVAREZ-GARCIA, Juan A.; SORIA-MORILLO, Luis M. deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. Neural Networks, 2018, 99: 158-165.

Name and workplace of master's thesis supervisor:

Ing. Jaroslav Bušek, Ph.D., U12110.3

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **31.10.2019** Deadline for master's thesis submission: **17.01.2020**

Assignment valid until: ____________

Ing. Jaroslav Bušek, Ph.D.		prof. Ing. Michael Valášek, DrSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

List of Abbreviations

ADAS	Advanced Driver Assistance Systems
AD	Autonomous Driving
ANN	Artificial neural Network
CNN	Convolutional Neural Network
GTSRB ,	German Traffic Sign Recognition Benchmark,
ROI	Region Of Interest

Table of Contents

List of Figures

List of Tables

1.0 Introduction

1.1 Background and Research

The last few decades have seen a immense acceleration in the acceptance of machine learning algorithms across an increasingly broad range of applications, many of which helps to simplify our daily tasks. Speech understanding, Spam filtering, face recognition, and e-commerce recommendations are only a few examples of applications in which machine learning methods are currently deployed.

In particular, with new developments of general-purpose computing on graphics processing units (GPGPU) and the availability of huge open data sets, training artificial neural networks (ANNs or NNs) with numerous hidden layers has become feasible. This approach to machine learning is known as deep learning, has revolutionized research in computer vision, speech recognition and Natural language processing(NLP), and is now swiftly being adopted in a large number of applications across a broad range of industrial sectors. Major technology companies such as Microsoft, Google, Facebook, Yahoo!, and IBM are currently re-thinking some of their core products and assistance to include deep learning based solutions. At the same time, hardware producers such as Mobileye, Nvidia, Altera, Qualcomm, and Samsung have started research and development projects with the aim of efficiently implementing deep neural networks on chips or field-programmable gate arrays (FPGAs), [1–3]. in order to set-up state-of-the-art vision systems in autonomous vehicles, smartphones, and robots.

In the automotive industry, machine learning algorithms are playing an major role in the development of advanced driver assistance systems (ADAS) for improving the driver's safety and comfort. Many automotive companies have now started considering deep learning as one of the most promising emerging technologies, not only in the improvement of ADAS, but also in the broader context of autonomous vehicles, constrained to revolutionize the entire automotive sector in the near future

In particular, systems for automatic traffic sign recognition, for example Toyota's Road Sign Assist (RSA), Volvo's Road Sign Information (RSI), or have already been on the market for a few years and are key factor's of modern ADASs. Improving the efficiency of both the detection and classification of traffic sign would boost the effectiveness of current systems, and potentially play an vital role in the development of future auto-pilot computer systems. There is therefore a solid interest among vehicle manufacturers to leverage recent developments in deep learning, namely deep Convolutional Neural Networks (CNNs), which have achieved state-of-the-art performance in a wide range of computer vision tasks such as image classification, localisation, and detection

2.0 Features of Autonomous Car

2.1 Self Driving Cars

Autonomous Cars are the Vehicles which are driven by latest Technologies without any Human interventions. They are able to navigate and driving themselves on to the roads by sensing the environmental impacts. They are designed in such a way that they occupy less space on the road in order to avoid traffic jams and to reduce the number of accidents on roads. The possibility of building a system able to drive a car has been Proposed already in the last century, but the technology available at that time wasn't able to solve such difficult task. Although the progression is colossal, in 2017,The automated cars which were allowed on public roads were not fully autonomous, every car needed a human driver who notices when it is decisive to take back the control over the vehicle[4].

Before to get notable results it too several years, Many giants in automotive industries like General motors, Ford, Mercedes Benz, Audi, Volkswagen, Toyota, Nissan, Volvo and BMW has started testing autonomous driving since 2013

Since around 2005 BMW has started testing Self autonomous systems, , Audi tested its self autonomous Audi TTS in 2010 by sending it to the top of pikes peak which was close to race speed. In 2011 GM made an self autonomous electric urban vehicle EN-V(short for Electric Networked Vehicle).. Volkswagen in 2012 began testing a "Temporary Auto Pilot" (TAP) system which allows a car to drive itself at speeds of up to 80 miles per hour on the highway. Ford has supervised extensive research into driverless systems and vehicular communication systems. Toyota in January 2013 manifested a partially self-driving car with numerous sensors and communication systems.

Tesla motors liberated the first version of its Tesla Autopilots in 2014 on board of Model S. Even if tender or younger than the other companies operating in automotive, Tesla is dominating the today market of self-driving cars ,and is working to establish a self-driving coast to coast drive from Los Angeles to New York. Waymo started as the Google self-driving car project in 2009. Today it is an sovereign company which produces one of the most advanced self-driving systems existing.

Waymo has invested billions in computer simulations and has ran its self autonomous cars over 5 million miles in 25 cities. The company is now planning to provide the first form of transit as a service using self-driving cars with a project called Autonomous Shuttle Service.

In addition to the big actors of the automotive industry, more and more start-ups have Experimented into the field of autonomous driving relying on new technologies and highly educated staff. Through the most successful and settled ones we have Zoox, Roadstar.ai, Pony.ai, Aurora, TuSimple, Drive.ai.

The autonomous vehicles moved from being a project hardly realisable to a reality of rising interest, which advance the research and the economic investments in the industry [5].

2.2 ADAS and AD

The terms ADAS (Advanced Driver Assistance Systems) and AD (Autonomous Driving) are often puzzled. The difference between the two can be blistered down to the level of human intervention required by the system in order to drive a car. Up to what level are drivers liable, and from what level can cars help us? This threshold splits between AD and ADAS. The goal of ADAS is to increase the number of situations in which encounters are avoidable. For each Particular ADAS feature, it is possible to measure the rate of accidents that can be avoided thank to it. In this sense, the evolution of ADAS can be noted as the process to eradicate the situations where encounters could occur by introducing various technologies.

There are at least 30 different types of ADAS features and they are in continuous evolution[6]. For example, many of them aim to prevent forward encounters, like Automatic Emergency Braking, Bicycle Detection, Forward Collision Warning , Obstacle detection ,Pedestrian detection, Others are used to control speed, like Adaptive Cruise Control or Curve Speed Warning, or to control the steering wheel angle, like Lane Keeping Assist or Lane centring, Lane Departure Warning are considered ADAS also those features which improve the approach of the driver through the sensors on board of the car, like Adaptive Headlights which moves and revolves the car front lights to keep them pointing the road, or the Automotive Night Vision, which uses thermo-graphic camera to increase driver's approach in darkness or poor weather conditions. Many of these features are shown in figure 1, where they are associated with the correspondent sensor used to implement them.

The Model of autonomous driving is a bit different. The machine o the vehicle takes care of a Exact part or all the tasks in restoration of human driver. Generally speaking, an AD system is expected to perform the whole heap of actions necessary to the driving action. This process involves the approach of the surrounding environment through the sensors, the amplification of the data collected which produces a control decision, and the definite actuation of the decided action.

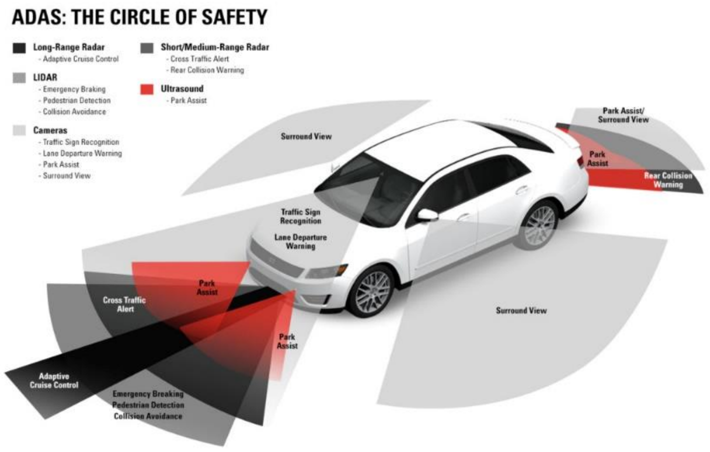


Figure 1:ADAS features and correspondent sensors used to develop them [7]

The level of automated driving depends on to what extent and/or in what background the tasks are automated. In order to classify the level of autonomy reached by an AD system, SAE (Society of Automotive Engineers) created a scale of 6 levels going from "no automation" to "full automation". These level are continued in table 2.1 released by SAE, and better explained here below:

- Level 0: Automated system issues warnings and may momentarily interfere but has no sustained vehicle control.

- Level 1 ("hands on"): The driver and the automated system split or share control of the vehicle. Examples are Adaptive Cruise Control (ACC), where the automated system controls speed; and Parking Assistance, and the driver controls steering, where steering is automated while speed is under human control. The driver must be ready to takeback full control at any time. Lane Keeping Assistance (LKA) Type II is a farther example of level 1 self driving.

- Level 2 ("hands off"): The automated system takes total control of the vehicle (accelerating, braking, and steering). The driver must control the driving and be prepared to interfere immediately at any time if the automated system fails to respond properly. The shorthand "hands off" is not meant to be taken actually. In fact, contact between hand and wheel is often compulsory during SAE 2 driving, to confirm that the driver is ready to interfere.

- Level 3 ("eyes off"): The driver can carefully turn their attention away from the driving tasks, e.g. the driver can watch a movie or text. The vehicle will handle situations that call for an urgent response, like emergency braking. The driver must still be prepared to interfere within some limited time, stated by the manufacturer when called upon by the vehicle to do so.

- Level 4 ("mind off"): As level 3, but no driver attention is ever required for assurance, i.e. the driver may go to sleep or leave the driver's seat. Self driving is supported only in limited geographical areas (geofenced) or under special conditions, like traffic jams. Outside of these areas or conditions, the vehicle must be able to safely interrupt the trip, i.e park the car, if the driver does not take back control.

- Level 5 ("steering wheel optional"): No human interference is required at all. For example - robotic taxi. As of today, none of the automotive companies can grant a fully autonomous system. At the moment, the convenient goal is to reach Level 3 of autonomy. As an example, the 2018 Audi A8 Luxury Sedan is the first commercial car to claim to be competent of level 3 self-driving. This individual car has a so-called Traffic Jam Pilot. When activated by the human driver, the system takes full control of all forms of driving in slow-moving traffic at up to 60 kilometres per hour[6].

Despite real Level 3 characteristics are assured, this function works only on highways with a physical barrier splitting one stream of traffic from oncoming traffic. The procedure to make Level 3 working on all the possible road conditions is still long, but the People who are investing more in autonomous driving industry are used to make very expectant predictions. for example, Tesla is planning to reach Level 4 of automation within 2020, but at the moment it is yet to reach level 3. Waymo was the first company to test Level 4 with passengers not employed in the company. The tests were successful, but this function worked only in an definite and limited area, within a 160 kilo-meter radius of Chandler, Arizona, a suburb of

Phoenix, where the accuracy of GPS is far above the average (few centimetres). In order to understand better which are the problematics experienced in the way to reach full autonomy, and to see which are the possible steps that can be taken to solve them.

2.1.1 Sensors

An autonomous system relies on primarily three functional blocks:

- Perception: sensing the surroundings and define a depiction of the environment.
- Decision Making: analyse the environment and conclude an action.
- Actuation: put the action in place.

Talking about sensors, we are dealing with the first of these three stages.
The perception systems of AD sensors are primely classified in two forms:

- Proprioceptive sensors (responsible for sensing of the vehicle's state like inertial measurement unit ,wheel encoders etc.)
- Exteroceptive sensors (responsible for sensing the ambient surrounding like cameras, RADARs LiDAR, ultrasonic, etc)

The exteroceptive sensors are the most significant in autonomous driving because they have to characterize the surrounding environment in a way detailed enough to allow correct decisions in the next stage. The main sensors belonging to this category are illustrated in the following sections.

2.1.2 Camera

Probably the most frequent and cheap sensors, cameras are available in all the models of autonomous cars and are used to implement many ADAS features. Camera-based systems are either stereo-vision or mono--vision, that is a set of multiple (Generally two) mono-vision cameras just like human eyesight. It is also possible to have four or six cameras pointing in the same direction in order to rebuild a three-dimensional image. Depending on the position where cameras are put they are used to develop different features, common places are the side mirrors, front grills, rear door and rear windshield.

Cameras are able to closely observe nearby vehicles, lane markings, speed signs, high-beam and notify the driver when the car is in danger of an imminent encounter with a pedestrian or an advancing vehicle.

Figure 2:Mobileye -market advanced driver assistance systems provide visual and audio alerts to drivers[8].

More advanced camera systems combine advanced software's which can not only detect objects, but detail them and regulate their trajectories. A big firm in the field of smart cameras is Mobileye, an Israeli company subsidiary of Intel which is working with many automakers, such as General Motors - for its Super Cruise system - Audi, Nissan, BMW, Fiat Chrysler, Honda, and China's Nio. From the beginning, Mobileye's ideology has been that if a human can drive a car based on vision alone, so can a computer. Meaning, cameras are critical to allow an automated system to reach human-level approach/actuation: there is an abundant amount of information (implicit and explicit) that only camera sensors with full 360- degree coverage can extract, making it the main component of any automotive sensing suite.

2.1.3. Radar

Radars (Radio Detection And Ranging) use radio waves to identify objects and determine their range, angle, and/or velocity. In AD application we find both long-range and short range RADARs (figure 3). Long-range radars can cover relatively long distances (around 200 meters) usually at high speed, while short-range radars work at slower speeds and they are used to sense the environment in the vicinity of the car (30 meters).

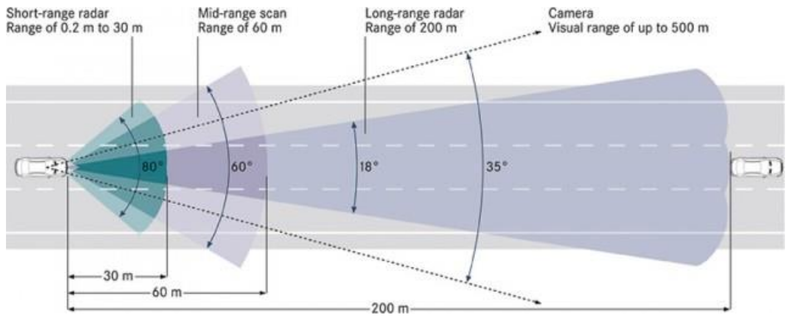


Figure 3: Different ranges covered by radars[9].

Radars can get information about closed objects, including their distance, size and speed. Exploiting radars, ADAS can notify the human driver in dangerous situations and depending on the system, they can even activate advanced steering or breaking. Short-range radars are used to implement lane-change assistant, Blind Spot Detection, collision avoidance, Cross-traffic monitoring, Park Assist, Automatic distance control and many other features. Radars high-precision and weather-agnostic capabilities make them a permanent fit for any autonomous vehicle prototype, not withstanding the ambient conditions[9].

2.1.4 Lidar

Lidars (Light Detection And Ranging) do the equivalent thing of Radars, but with pulsed laser light rather than radio waves. Differently, than radars, Lidars do not simply identify the objects but describe them by illuminating the surroundings and analysing the path of the reflected light. This operation is repeated over a million times per second, yielding to a 3D high-resolution cloud point representing the surroundings (figure 4). Given that Lidars are using transmit light, they are robust against various light conditions, during all the times of the day and with any weather condition.

All these superiority make Lidars the preferred technology of many industries running in the field of AD, starting from Waymo, which produces his own sensors in-house.

However, there are also some downsides of lidars. For example, they are very huge and exposed in peripherals positions like the roof of the vehicles, which makes them ordinary and easily damageable (figure 5). Anyway, the main drawback is probably the price. Lidars are generally far more costly than radars and cameras; to have an idea, at the beginning of their

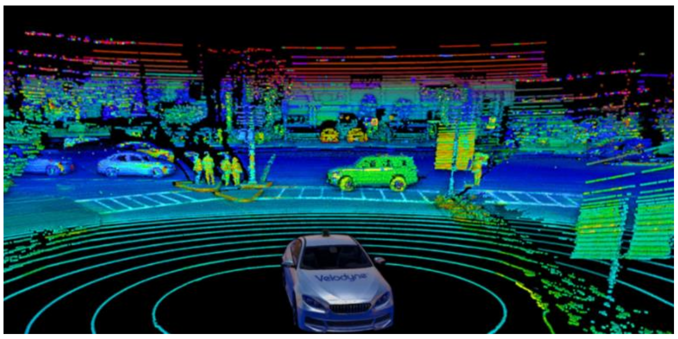


Figure 4: Lidar point cloud representing the surroundings of a car[10]

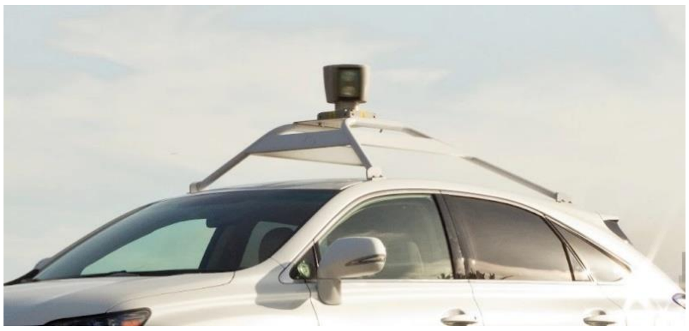


Figure 5 : A Lidar mounted on top of a car to maximize the range of action[11]

commercialisation for the autonomous car market, the finest models of Lidars developed by Velodyne cost 75.000$

.

Today there are low-cost versions advertised by the same company at a price of 7.999$. Given the remarkable description capacity, much money is being invested in research in order to obtain an even low-cost version. For example, a technology called Solid State Lidar seems to be very encouraging and would eventually allow to break down the price by a factor of ten[14][15].

2.2 Image processing for AD

Despite the drawbacks mentioned in the previous paragraph, the quality of data provided by Lidars make them the superior choice in the strategy of most of the autonomous driving industries (Daimler, waymo, Bosch, Audi General Motors and others). Following a different strategy, Tesla decided to do not mount any Lidar sensor on its cars. The Waymo's self-driving minivans use three various type of Lidars, eight cameras and five sensors while Tesla cars mount twelve ultrasonic sensors, eight cameras and one forward-facing radar.

The Tesla CEO Elon Musk is firmly convinced that Lidar are unnecessary to develop fully autonomous cars;
"In my view, it is a crutch that will drive companies to a local maximum that they will find very hard to get out of... Perhaps I am wrong, and I will look like a fool. But I am quite certain that I am not."[12].

Tesla focuses its intensions and hopes in camera sensors combined with radars and ultrasonic sensors. Without a doubt, this is a perilous choice, in fact, the company is operating across a completely various path compared to the other competitors; despite the steady investments, for the juncture Tesla does not excel for the autonomous driving capabilities provided. On the other side, if Tesla should ultimately succeed in its intent, all the other companies could not abide to the competition rising from the cheaper costs of the cars and would be compelled to start working in the same direction. The idea behind this planning is that camera sensors provide enough data to describe the ambience and are consequently enough to allow a car to drive itself. Generally speaking, it is tough to contest this strategy too much. Human beings are perfectly able to drive cars without the need of radio, ultrasonic or light sensor; moreover, when they fail, most of the times is because of not of information and lack of attention,. If we already had the necessary knowledge in the field of computer vision, a few camera sensors would be everything we need to develop self-driving-cars.

Putting it in this way, it looks like it is only a matter of time before we will be able to get rid of sensors other than cameras. With enough progress in computer vision, one day the whole intricacy of self-driving-cars could reside entirely in the software, and the hardware needed for the computation, without the need of relying on more complicated and expensive sensors.

2.3 Lane Detection

A common technique often used in Lane Keeping or Lane Departure Warning Assist is Lane Detection. The lane is detected through a workflow which aims to classify the lane markers on the road. Doing this with a pure computer vision approach is not irrelevant. The most used algorithm is the one shown in figure 6 which follows the steps here below:

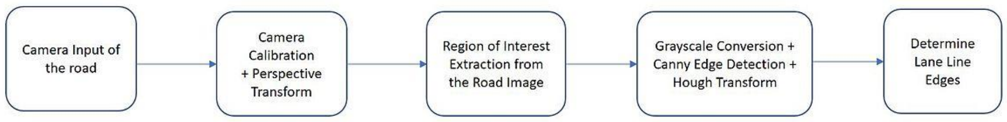

Camera Input of the road
Camera Calibration + Perspective Transform
Region of Interest Extraction from the Road Image
Grayscale Conversion + Canny Edge Detection + Hough Transform
Determine Lane Line Edges

Figure 6: The lane detection system uses the principle of Hough transform and Canny edge detector to detect lane line form real-time camera images(Image is taken from Wikipedia)

- Lane Departure Warning: it detects the lane markers and warns the driver if the car leaves the lane.

- Lane Keeping assist: it finds or detects the lane markers and takes control of the steering wheel in the case that the car leaves the lane, bringing it back to a secure position. After this action, the control is given back to the driver.

- Perspective Transform: a significant problem come across in image analysis is the deformation due to perspective. Indeed, images are a 2-Drepresentation of a 3-D space, thus some information loss occurs. The perspective transform step place any image in a bird-like view (from the prospect of a bird which observes the environment from the top); with this transformation, the lanes that are converging with the perspective will become correlate.

- Region of Interest: as the road is commonly located in the lower part of the image; it is viable to crop the input data and select only the lower part of the images.

- Canny edge detection: a series of steps are used to eradicate the noise from the image and detect the strongest edges within it. The implementation of this algorithm is not irrelevant, but there are various implementations present within libraries like opencv in python.

- Hough Transform: this technique takes as an input the edges identified in the previous phase, and uses a voting system to find the best nominee with a determined shape. In lane detection, we expect to find all the edges grouped over the shape of a line with the inclination of the lane marker, but likely that the edge detection is probably not perfect, equation of a line cannot be computed in common way. The goal of the Hough transform is to address this issue by grouping edge points into object nominees or candidates through an explicit voting procedure over a set of parameterised image objects. As for canny edge detector, also Hough Transform can be implemented within opencv in python.

The workflow described above can be considered as state of the art for lane detection. There are so many implementations available online as open source code as well as working tests of sample images representing real roads[13].

3.0 Background Theories

3.1 Artificial neural network

Artificial neural networks (ANN) are computing structures inspired by biological neural networks that constitute human or animal brains. In the most general form, an ANN is a system invented to model the way in which a brain performs a particular task. Such systems learn to do tasks by seeing given examples, generally without task specific programming. They have found most use in tasks where it is impossible to apply rule-based programming [16].
This makes an ANN a good successor for human detection, since it is barely possible to manually program a classifier which will consider all available positions and all different appearances of humans.

An ANN is based on a collection of associated units called artificial neurons (analogous to axons in biological brain). Each connection (called synapse) between neurons can address a signal to another neuron. The receiving neuron can alter an input signal and then send a signal to another neuron.

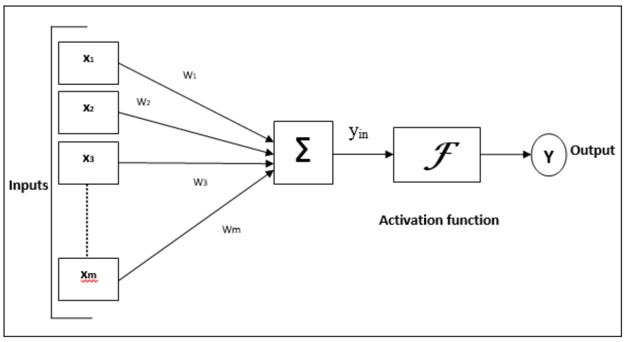


Figure 7: Model of an artificial neuron.[17]

Basically, ANNs are made up of a set of artificial neurons, an artificial neuron has n number of inputs, the inputs in the artificial neuron are similar to the dendrites in a biological neurons. Whereas the output of an artificial neuron is similar to the axon of biological neuron, Each input $X_1, X_2, X_3 \ldots\ldots X_m$ has an assigned weight $w_1, w_2, w_3 \ldots\ldots w_m$. Weighted input are formed in single unit and run through an activation function producing some output y as shown in Figure 7. The network is formed by joining the neuron output with the input of an another neuron. ANN is therefore effectively described in figure 7

$$Y = \sum_{j=1}^{m} w_{mj} * X_j. \tag{3.1}$$

The output of the neuron Y, would therefore be the outcome of some activation function on the value of y_m

Note: It is not mandatory that the output of the Neuron Y is going to be only one. It can be Binary, Categorical or Continuous.

- Binary-The output can be binary if the we get the Value either in 0 or 1.for example: detecting a pedestrian, if there is no pedestrian on road then the output will be zero and for vice versa case the output will be 1.
- Categorical Output: for example Classification of traffic signals.
- Continuous: We can use this method in ANN to perform regression.

3.1.1 ANN Advantages

- Generalization capability.
- Capable of modelling any relationship between a set of inputs and outputs.
- Forecasting

3.1.2 ANN Limitations

- Training time is too long for deep networks, which are most exact architecture for most problems. This is especially correct if we are training on CPU instead of GPU
- Lots of data required, mainly for architecture with many layers. This is bit problematic for most ML algorithms and on the other side it is relevant for ANNs because of large number of weights and connections.
- For best results, architecture have to be fine tuned, there are many design decisions are needed, from layers to nodes in each layer to the activation functions

3.1.3 ANN Applications

- Autonomous Cars
- Image classification
- Fault Detection and isolation
- Face and speech recognition
- System Identification and control Systems application
- Finger Print recognition
- Handwritten Character recognition
- Business applications such as bank failure prognosis and stock market predictions

3.2 Unit Step(threshold).

A unit Step or binary step function is threshold-based activation function. The neuron is activated and send accurately the same signal to the next layer, if the input value is above or below a certain threshold[18]

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

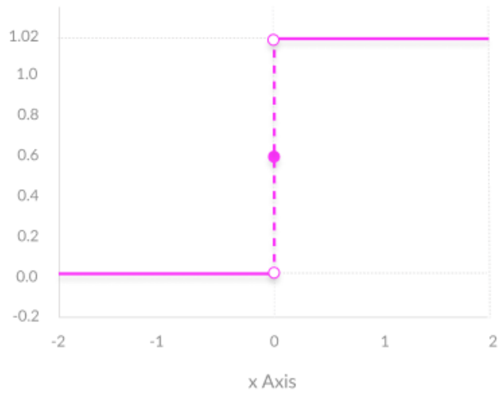


Figure 8:Binary step function [18]

3.3 Activation Functions

There are many activation functions that is used to change the output of our neuron.

Non-linearity in the network is introduced by Non-linear activation functions. Without a non-linear activation function, the network can only read functions which are linear combinations of its inputs. Using the sigmoid function in the output layer as activation function is very common practice. Its expression is

$$f(x) = \frac{1}{1+e^{-x}}. \quad (3.3)$$

1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4
ReLU
5
4
3
2
1
0
−4
−2
0
2
4

Figure 9: On the left: sigmoid activation function   On the right: rectified linear[19]

The rectified linear function is generally used for hidden layers [18]. It is a simple non-linearity: It evaluates to 0 for negative inputs, and positive values remain unaffected (f(x) = max(0,x)). The gradient for the rectified linear function is 1 for all positive values and 0 for negative values. It means that during backpropagation, negative gradients will not be used to restore the weights of the outgoing rectified linear unit. However, because the network has a gradient of 1 for any positive value it has much advance training speed when compared to other non-linear functions due to the good gradient flow. According to [20], deep convolutional neural networks with ReLUs train many times faster than their equivalents with tanh units. For example, the logistic sigmoid function has very slight gradients for large positive and negative values so that learning nearly breaks in these regions (this behavior is similar to a saddle point). The ReLUs also have the desirable property that they do not need input normalization to avoid them from saturating [20]. In fact, despite the fact that negative gradients don't propagate with rectified linear functions (the gradient is zero here), large gradients for positive values are very caapable and ensure fast training without regard of the size of the gradient.

3.4 Multilayer Perceptron

A Multilayer is a definite acyclic group the nodes are neurons with logistic activation. Neurons of i^{th} serve as input features for neurons of i+1^{th} layer. Highly complex functions can be calculated combining many neurons. The nodes with no target of any connections are called input neurons. If a MLP is applied to input pattern of dimension then it must have n input neurons, one for each dimension, input neurons are generally titled as neuron ,neuron 2,…..
The nodes without the source of any connections are called output neurons. The desired Value of the Training pattern decides the number of output neurons. The nodes which does not have any input or output are called Hidden layer.as the graph is acyclic, all neurons can be organized in layer with the input as being the first layer

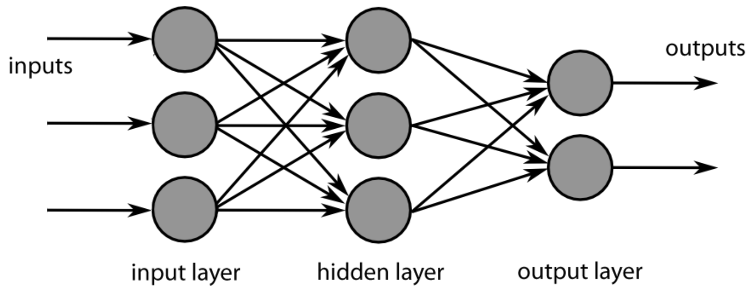


Figure 10: Multi Layer Neural Network [21]

CNN can choose window (Proposal) feature extraction and classification. It comes from the traditional full connection Neural Network model development and to the depth of the Network model, that is one of the important algorithm is widely used in computer vision field. The main feature of CNN is that it draws on the concept of human visual perception field and proposes the concept of weight sharing. The traditional neural network model is fully connected, that is, each neuron in each layer of the network is connected to all neurons in the upper and lower layers, so if the number of neurons in the two adjacent layers are M and N respectively, then only these two neurons are connected.

3.5.Deep learning

The design and development of algorithms letting a machine to learn from huge amounts of data and make forecasts about the future is critically[22]. reliant on the process of extracting the most informative and non-superfluous information from such data in their raw form. This process of feature extraction is commonly carried out by humans and requires in general a great deal of expertise and activity, often including trial and error methods. Increasingly, this process is being replaced by representation learning, a set of methods to naturally learn and discover new representations of the raw data, often surpassing traditional hand-crafted feature extraction

3.6 Deep neural networks

Deep learning, a set of machine learning methods typically based on artificial neural networks (ANNs), is proving very successful in a wide range of tasks, particularly in the background of supervised learning, i.e. assuming a function outline given input data to desired given output values, in order to map new, lurking examples. Deep neural networks alter from other machine learning algorithms in that the depiction of features is organized over multiple layers of nonlinear processing units, which transform the depiction or representation at one level into a somewhat more abstract representation at a higher level. Typically, the first layers of a deep NN discover very uncomplicated features of the data (e.g. colour gradients or edges, if the data has the form of an image). More and more abstract and complicated features are then naturally learned downstream in the network, by sooner or later building them out of lower level ones, creating a ranking of representations.

One key idea behind the stability of neural networks is that these hierarchically arranged features can to some extent be learned individually of one another, in a distributed fashion. A distributed representation of features is a vastly desirable property in machine learning when dealing with high-dimensional data. If a d-dimensional input space (e.g. $\mathbb{R}^d$), a distributed portrayal of n features can divide the space into O(n^d) regions, equivalent to intersection of half-spaces. If such regions represent concepts, then exponentially many concepts can be esteemed using such a representation, since the different attributes of these concepts are shared [23].

It is noteworthy that models of artificial neural networks have been familiar and used for decades. Pioneering work in neural computation dates back to 1943, with the introduction of McCulloch-Pitts neurons [24], simple threshold units with binary input and The first feedforward neural network (FFNN) with flexible synaptic weights, called as the perceptron, was introduced by Rosenblatt [25] in 1958 and used to frame linear classifier machines with simple adaptive capabilities. A breakthrough was the analysis that multilayer FFNNs can be trained using an algorithm known as backpropagation to figure out the gradient of some objective function with respect to all model parameters, and then simply implement gradient descent to update such parameters (see Aside 2.1.1). This discovery is now credited to Werbos [26]

In 1989, Hornik [27] demonstrated that an FFNN with as less as a single hidden layer can approximate any function to any desired degree of accuracy. Due to this fundamental theoretical result, known as the universal approximation theorem, the whole concept of training neural networks with a huge number of hidden layers may not sound as the optimal approach, as it arguably increments the complexity of the model. However, it was thereafter discovered, mainly empirically, that approximating complex functions using deeper models can be much more useful in terms of the total number of hidden units in the model. compositionality, Intuitively, i.e. the possibility of creating greater level features by combining lower level features, adds another exponential asset to the statistical efficiency of a neural network, on top of the exponential asset given by the distributed nature of the represented features. This intuition is progressively supported by theoretical results. For example, Delalleau [28] have demonstrated that functions in a specific class of polynomials in n variables can be represented by sum and product networks of depth O(k) using O(nk) hidden units, but require O($(n-1)^k$)) units in the case of a 1-hidden layer network.

3.7 Convolutional neural networks

Convolutional in mathematics[46] is an operator, refers to the mathematical combination of two functions to produce the third functionA typical case is a colour (RGB) image, which is essentially a heap of three 2D arrays, and it can be viewed as a 3D tensor, with the third dimension being the colour channel. The architecture of CNNs was first seen in the 1980s, with Fukushima's neocognitron [29], inspired by a neurophysiological model of mammal's visual primary cortex introduced by Hubel and Wiesel [30] in the 1960s. Many essential aspects of this early model are still used by modern CNNs. At its core, a convolutional neural network is a heap of layers transforming a 3D input tensor to a 3D output tensor in a feedforward fashion. These layers implement linear or nonlinear transformation of their input, and these operations may or may not hold additional parameters and hyperparameters. Although this characterization may arguably apply to any form of FFNN, convolutional neural networks are peculiar in that the extraction of features in the first layers, as well as the architecture of such features into higher-level features, is implemented through mathematical operations called discrete convolutions.

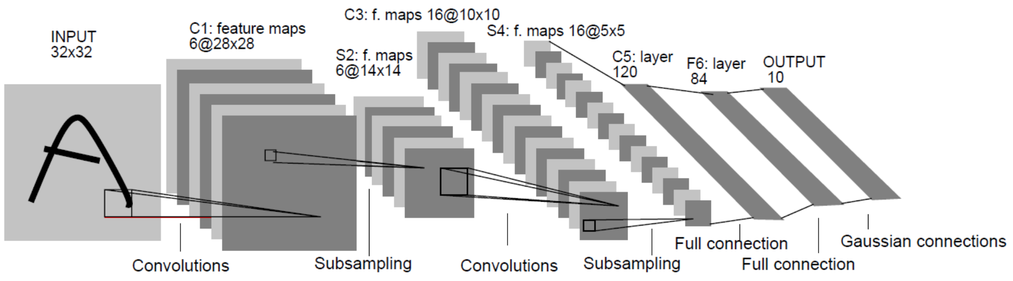


Figure 11: The architecture of LeNet-5, a CNN used by LeCun [32] for document reading.

First time Convolutional neural networks were trained with backpropagation and gradient descent by LeCun [31] in 1989, for the intent of classifying handwritten digits. In the 1990s, CNNs-based algorithms were already used in economic applications, such as reading cheques [32] (the CNN architecture called as "LeNet-5" used in this work is shown in Figure 11). However, at that time the calculation power needed to train larger and deeper models to solve more complicated tasks was simply too large. This aspect, together with the fast progress made in other areas of machine learning, hindered the dispersion of neural networks in the computer vision community for almost two decades. A breakthrough result was published in 2006 by Hinton [33], introduced an algorithm for greedy layer-wise pre-training algorithm to accurately train particular kinds of deep NNs, and broadely contributed in reviving the interest in deep learning. However, its recent success would have been impossible without two other essential ingredients. First, the appearance of open,large and labelled data sets, such as the popular CIFAR10 and ImageNet data sets of natural images. Secondly, the dispersion of cheap, massively parallel computing power in the form of GPU's (Graphics Processing Units) for normal purpose computing, starting from the late 2000s. Human-level performance in handwritten letters and digit recognition on the MNIST database was achieved for the first time in 2011 by Ciresan [34] by using a GPUtrained deep CNN.Most famously, in 2012 the ImageNet (ILSVRC) contest, the largest contest in object recognition with 1.2 million high-resolution images in 1000 classes was won by Krizhevsky [1] using a deep CNN (now known as "AlexNet"). By putting this result into perspective, they attain a top-5 classification error1 of 15.3% on the test data, almost halving the second-best entry in the same competition (26.2%).

3.8 Convolutional layer

A CNN consists of several convolutional layers which are used to extract features from the input of the network. One of the major concept of CNNs is that the same conversion is applied at alllocations. The filter, or convolution, applies to a specific neighbourhood of nodes (pixels) as seen in Figure 12

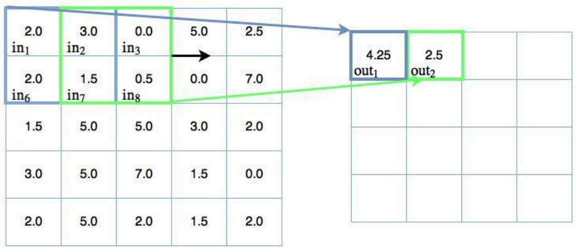


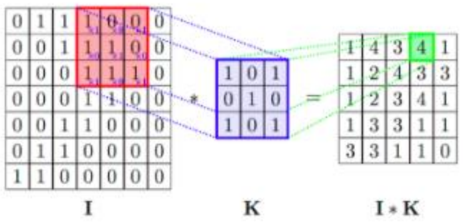
Figure 12 :Example of max pooling where the max is taken over 4 number

Important to notice is that the filter weight values, w_{ii}, are parameters that are to be tuned and need not all to be equal as in the example (Figure 12) where all the weights are 0.5

The use of a filter means supposing that pixels which are spatially nearer together, more likely than pixels on opposite corners, will collaborate on forming specific features of The spatial location of the feature is always remembered at least as far as the second fully connected layer, at which point whether it is necessary or not is determined by the weights of that layer [35].

Given a 2D-image below, I, and a small matrix, K of size h x w, the convolved image can be figured out, see Figure 13. The matrix K is the convolution kernel, which is assumed to encode a way of extracting an interesting image feature. The convolved image, I * K, is computed by overlaying the kernel over the image in all possible ways, and at the same time taking the sum of element-wise products between the image and the kernel, equation 3.8. Also, to make the equation complete a non-linearity (ReLU) should be added.

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} = K_{ij} * I_{x+i-1,y+j-1} \qquad (3.8)$$



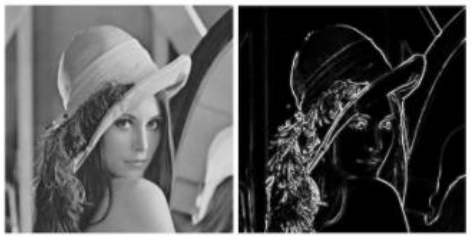


Figure 13: The result of applying convolution (with two separate kernels) over an image, to act asan edge detector. Matrix I represents the 2D image and the small matrix K the filter and I * K represents the convolved image [36].

An important thing to notice is that not all input nodes are connected to output nodes. Another thing is that as the filter moves around, the same weights are being applied to the whole image. The application of the weights results in a transformation of the image[36]. The weights are parameters that is adjusted during training but during calculations the weights are constant. The weights within the filter could be any combination of values depending on how the filter is trained. Compared to fully connected neural networks, these two features of CNNs can significantly reduce the number of parameters required. If the number of parameters are reduced the complexity of the model will be reduced and also the expected error. CNN reduces the complexity by having shared weights and having not fully connected layers. After the convolutional mapping the output is passed to a non-linear activation function, often the rectified linear unit activation function since it handles the vanishing gradient problem. This is called feature mapping. The fact that the weights are held constant as the filter moves allows the filter to be trained to recognize certain features in the input data. In the case of images, it may learn to recognize shapes such as lines, edges and other distinctive forms. However, in order to classify well many filters are needed.

3.9 Pooling Layer

The output feature map can be sensitive to the location of the features in the input. This sensitivity can be addressed by downs ampling the feature maps. The invariance of feature detection sensitive to the location is refereed by the technical phrase "local translation invariance "[47]. Hence making the feature maps a more robust to change in position of the feature in the image pooling is performed. Common usage of the pooling layer is to down sample, there are no trainable parameters associated with the pooling layer. The pooling can be performed by averaging or taking the max of the features in the patch of the feature map. Some common methods of pooling are Average pooling and Max pooling. Max pooling can be seen by figure 14

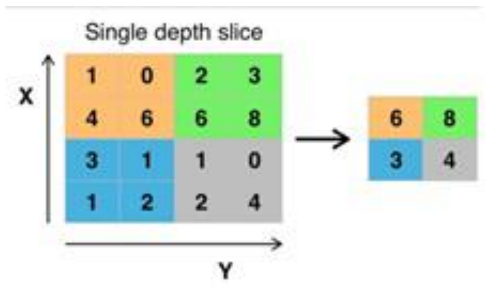


Figure 14: Example of max pooling where the max is taken over 4 number with stride 2 [37]

3.10 Back Propagation

From figure 14, loss function E depends only on w_{ji} and a_j

Error δ_j is given by

$$\delta_j = \frac{\partial E}{\partial a_j} \tag{3.10.1}$$

and δj is the error of the neuron on the output of hidden layer and z_i is the input from i to j and known form forward pass.

For output layer δk depends only on ak via g(ak) and can be written as:

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial \hat{y}k}\frac{\partial \hat{y}k}{\partial a_k} = g'(a_k)\ \frac{\partial \hat{y}k}{\partial a_k}\text{ [38]} \tag{3.10.2}$$

and for hidden layer δj E depends on ai via all ak and computed as:

$$\delta_j = g'(a_j)\sum_{k\in Dest(j)} w_{k_j}\,\delta_k\text{ [38]} \tag{3.10.3}$$

Hence the derivative $\frac{\partial E}{\partial wd}$ can be computed with

$$\frac{\partial E}{\partial wji} = \delta_{jzi}\text{[38]} \tag{3.10.4}$$

The training is performed repeatedly to reduce error by minimizing the loss function. In each iteration, the weights get changed to improve its performance. Often a large data set is required to train the neural network. Initially, the network starts with some random number as the results from the forward propagation. These results from forward propagation can have error. The measure of error is found with the loss or cost. This measure of error is done with the loss function on the desired output and prediction of training examples. The learning process should be efficient to change the required network parameter to reduce the loss (error). Consequently, the negative gradient of the loss with respect to parameters is calculated by recursively applying the chain rule layer by layer towards the input. This process is repeated for each example, and the parameter learning rate of the obtained negative gradient is summed up to weight and update them. This process is called backpropagation. The learning algorithm must be optimized enough(should have proper value for parameter like learning rate and batch size so the algorithm do not stuck) such as stochastic gradient descent (SGD) [44] to get all parameters to converge to at least local minima.

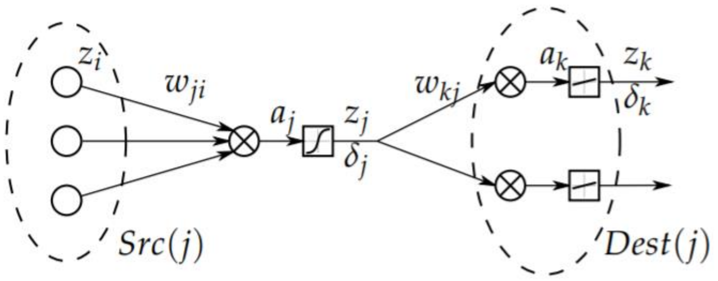


Figure 15: Backpropagation error[38]

We know that there's a process called Forward propagation where information is entered into the input layer and then it's propagated forward to get our output values and then we compare those to the actual values that we have in our training set. And then we calculate the errors then the errors are back propagated through the network in the opposite direction and that allows us to train the network by adjusting the weights. So one important thing to remember here is that back propagation is an advanced algorithm driven by very interesting and sophisticated mathematics which allows us to adjust the weights. All of them at the same time all the weights are adjusted simultaneously. So if we were doing this manually or if we're coming up a very different type of algorithm than Even if we calculated the error and then we were trying to understand what effect each of the weights has on the error we'd have to somehow adjust each of the weights independent independently or individually. The huge advantage of back propagation and it's a key thing to remember is that during the process of back propagation simply because of the way the algorithm is structured. We are able to adjust all the way at the same time so we basically know which part of the error each of our weights in the neural network is responsible for. Now that is the key fundamental underlying principle of back propagation. And this was why it picked up so rapidly in the 1980s and this was a major breakthrough[42].

3.11 Dropout

Dropout is a newly published regularisation method [43] that attempts to imitate model averaging with very large ensembles, with nominal computational impact. This is accomplish by de-activating (dropping out) each unit in a hidden layer of a network at each training iteration, with some probability p (generally 0.5), as illustrated in Figure 16. This simple procedure is equivalent to training an exponentially large ensemble of thinned sub-networks (namely 2NH models, where NH is the number of hidden units). dropout is not applied while testing, so that its predictions approximate the averaged predictions of the ensemble of its sub-networks.

Dropout is not exactly identical to model averaging, as thinned sub-networks share parameters and as such are not independent. Moreover, dropout usually slows down the training process, as a large fraction (typically 1=2) of the parameters are adequately frozen at each iteration. However, dropout is still computationally much competitive than training several models to average over, and it can be applied to nearly any type of model and with any training algorithm.

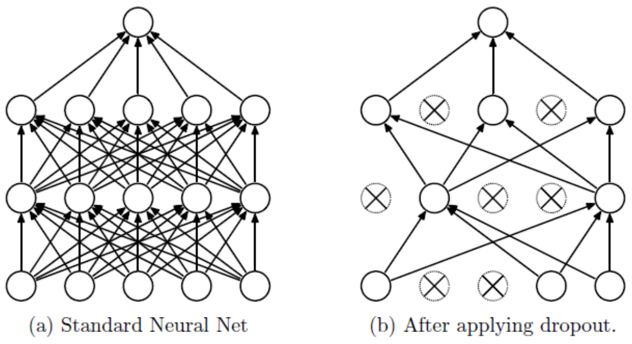


Figure 16: Graphical representation of dropout [43].

3.12 Softmax

It is generally use to perform multiclass classification,it assures that all the activation in single layer are summed up to 1.The softmax activation function is given in equation 3.12 ,where K is number of activation function,z is the vector of activation [40]

$$\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}} \quad \text{for j =1…..K} \qquad 3.12$$

The Relu activation function turns to positive part of its argument, omitting all negative inputs to zero

4.0 Software Tools

4.1 Deep learning using CNNs with Keras and TensorFlow

Keras [39] is a high-level neural networks API, written in Python and capable of running on top of TensorFlow [24] TensorFlow[24] is a machine learning computational backend developed by Google LLC. It is a tool for expressing and implementing a machine learning algorithm. Tensorflow library consists of visualization tools like TensorBoard, which make debugging and optimization of the neural network easier. The computation developed in TensorFlow can be executed in a variety of systems with little or no change making it a flexible tool.

4.2 Google Colab

It is a free research tool for machine learning education and research tool which uses Jupyter [notebook environment from Google LLC, which can be run a maximum of 12 hours in single runtime[44]. Google Colab has support for deep learning application like Keras, TensorFlow, Pytorch and OpenCV Also, the other dependencies could be easily installed using pip installer. It also allows three different kinds of runtime hardware like GPU, CPU, and TPU, which can accelerate the process.

5.0 Thesis Outline

5.1 Work performed in this thesis

In this thesis we use CNN for Traffic Sign Classification from images. The standard German Traffic Sign Benchmark Dataset (GTSRB) is used for testing and training. Given the objective of this thesis, we will focus entirely on the Classification of the German Traffic Sign Benchmark Dataset GTSRB dataset for the sake of obtaining high certainty and preprocessing technique, network architectures and parameters.

For the first Part the tasks that were carried out in this project are the following:
1. Implementation of a basic CNN architecture.
2. Training CNN classifier.
3. Preprocessing the training and testing data.
4. Tests on different settings and their effect on the performance.
5. Gathering the results and drawing conclusions.

5.2 Hyperparameters

It takes a lot of time for the Training of CNN or it might even take days, The time consumption depends upon the complexity of data and network. While designing an architecture to provide good accurate result multiple parameters are tuned while keeping the training time intact. The hyperparameters are tuned progressively are as follows.

Related to the training images:
- Size of image.
- Preprocessing parameters.
- Dataset size.

Related to the training:
- Number of epochs.
- Learning rate.
- Weight update criterion.
- Training loss expression.
- Batch size.

Related to the architecture:
- Type of layers.
- Number of layers.
- Non-linearities.
- Number of filters.
- Zero padding.
- Dropout.
- Strides.

The Second Part will be focused on Behavioural Cloning This includes deep neural networks feature extraction with convolutional neural networks as well as continuous regression. In summary we are essentially going to be downloading a self-driving car simulator provided to us Open-Source by Udacity. We are then going to use the simulator to create our very own training data for our model by driving a car through the training track inside the simulator as we drive the car through the simulator. We are going to be taking images at each instance of the drive. These images are going to represent our training dataset and the label for each specific image is going to be the steering angle of the car at that specific instance. We will then show all of these images to our convolutional neural network and allow it to learn how to drive autonomously by learning from our behaviour as the manual driver this main variable that our model will learn to adjust is the steering angle of the car at any given instance it will effectively learn to adjust the steering angle to an appropriate degree based on the situation that it finds itself in. Now after we train our model we are going to evaluate its performance on a completely different testing track where the car will be made to run autonomously if we are able to train the car properly it will perform very well on our second track and will drive on its own This behavioural cloning technique is incredibly useful and plays a vital role in real life self-driving cars as well. Cars are typically driven around and trained on real roads by manual drivers and they are then trained on the data that they collected on this drive to then clone the behaviour of their manual drivers. Therefore after learning this behavioural cloning technique you will effectively be able to understand and even apply the science of self-driving cars. Now this technique is not simple and involves complex deep learning techniques as well as image manipulation.

5.3 Method

5.3.1 The GTSRB data set

The German Traffic Sign Recognition Benchmark (GTSRB) is a openly available data set containing 51839 images of German road signs, distributed into 43 classes [45]. A representative image for every class is shown in Figure 17. The data set was published during a competition held at the 2011 International Joint Conference on Neural Networks (IJCNN).

Images in the data set shows large variations in terms of shape and colour among some classes, as well as capable similarities among others (e.g. different speed limit signs)[45]. Traffic sign recognition (TSR) is generally a multi-class classification challenge which requires to deal with unbalanced class frequencies. For Example, it is simple to understand that a 50 km/h speed limit is more frequent than a 120 km/h one. Adding to this a TSR dataset should account for situations like varying light changes, rotations, partial occlusions, weather conditions etc. It consists of vastly uneven classes in terms of number of samples for each class. The images follow the robust variations in visual appearance of signs due to illumination, distance, weather, partial occlusions, and rotations [46].

while training classification algorithms on the GTSRB data the additional challenge is that the 43 classes are not equally represented. for classes 0,19 and 37 the relative frequency is only 0.56% which is lower than the mean 1/43=2.33% (Figure 18, right panel). [45]

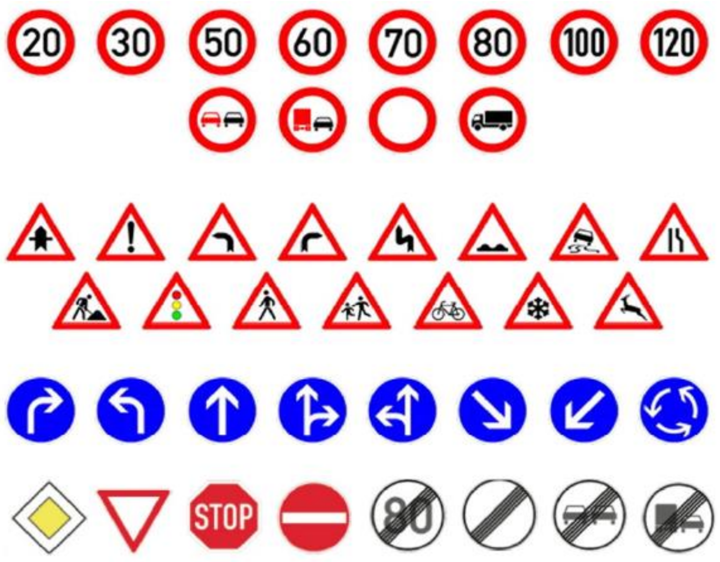


Figure 17:All 43 benchmark classes in GTSRB[47]

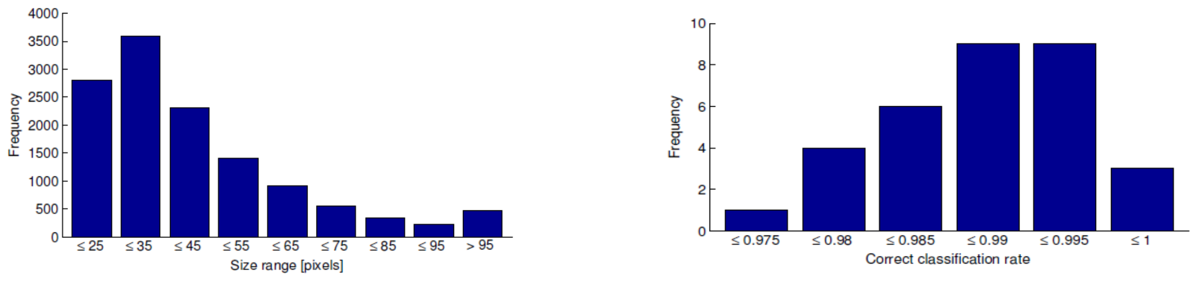


Figure 18 : Left panel: distribution of the resolution of GTSRB test images. Right panel: relative class frequency in the training data. Both images reproduced with permission from Stallkamp . [45].

5.3.2 Data Preprocessing

In accordance with the idea behind representation learning, GTSRB images were minimally pre-processed, and used to train CNNs almost in their raw form. The preprocessing steps described follows the approach used by Sermanet and LeCun [5] as closely as possible, for a better comparison. The data preprocessing is applied based on three steps. First, the borders of all the traffic signs are removed according to the annotations. And then, based on the resolution distribution of the GTSRB, all the cropped traffic signs are resized to 32×32×3 pixels. In other wards, the traffic signs are processed to have square bounding boxes. Finally, the colour normalisation of the traffic signs is utilised by linearly scaling all the pixels.

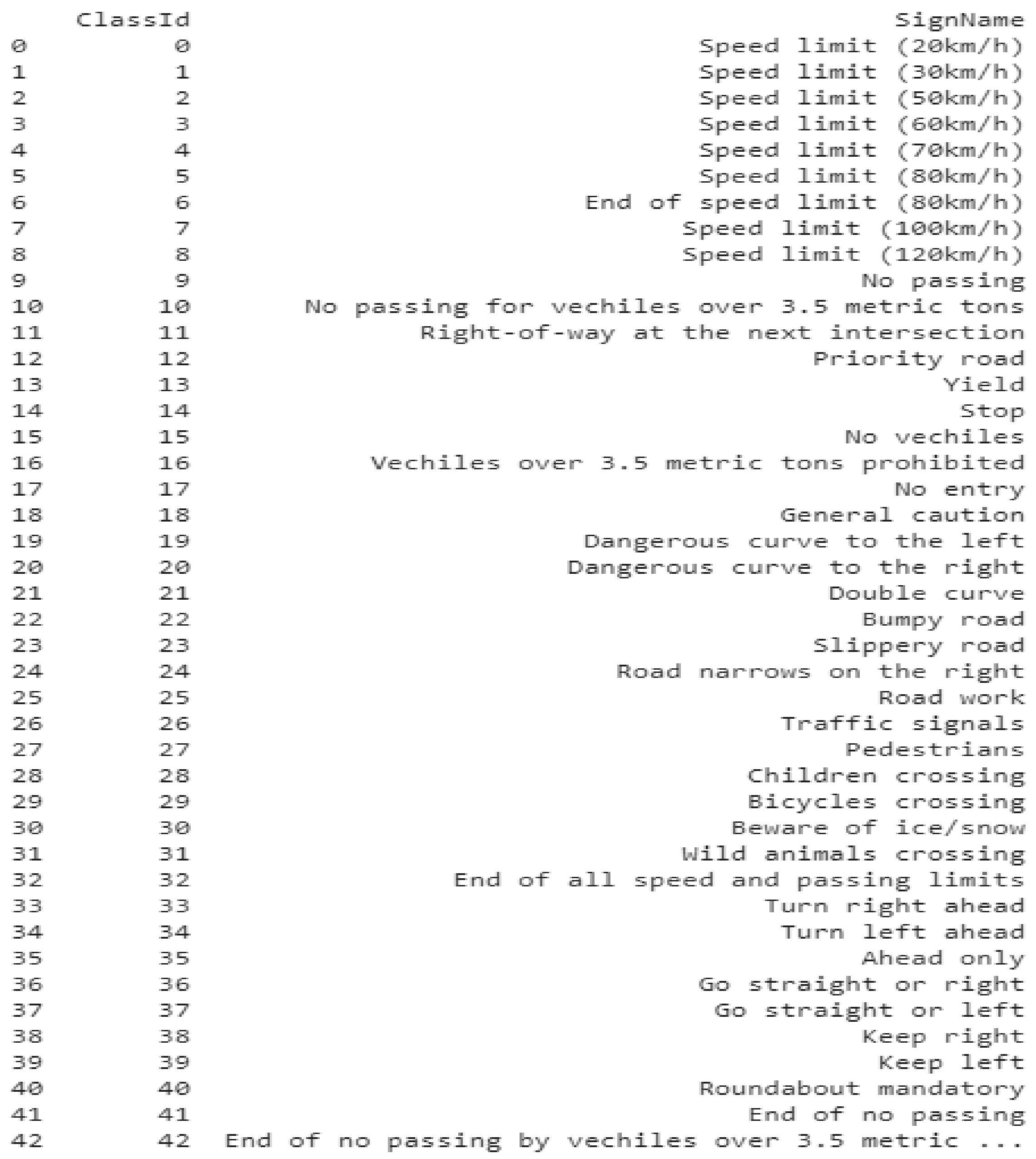
5.3.3 Performance Evaluation

The results of the experiments will be presented in detail in this section. The comparative experiment is implemented based on LeNet network architectures, regularisation strategies and optimisation algorithms. For the given experiment, data argumentation is not applied, and the generalisation ability is imposed via network architectures and some other regularisation strategies, dropout. In order to achieve more accurate results, all the experiments with same setups are utilised for 10 times, and the experiments with the best performance will be presented. Except as otherwise indicated, all the training configurations are the same as Table.

6.0 Results

6.1 System Environment

In order to implement all the TSR systems, a desktop with an Intel 2.5GHz CPU, 08GB memory and a NVIDIA Geforce GTX 1050, All the programs run on a 64-bit Windows 10 operating system.



```
    ClassId                                           SignName
0         0                               Speed limit (20km/h)
1         1                               Speed limit (30km/h)
2         2                               Speed limit (50km/h)
3         3                               Speed limit (60km/h)
4         4                               Speed limit (70km/h)
5         5                               Speed limit (80km/h)
6         6                        End of speed limit (80km/h)
7         7                              Speed limit (100km/h)
8         8                              Speed limit (120km/h)
9         9                                         No passing
10       10       No passing for vechiles over 3.5 metric tons
11       11              Right-of-way at the next intersection
12       12                                      Priority road
13       13                                              Yield
14       14                                               Stop
15       15                                        No vechiles
16       16           Vechiles over 3.5 metric tons prohibited
17       17                                           No entry
18       18                                    General caution
19       19                        Dangerous curve to the left
20       20                       Dangerous curve to the right
21       21                                       Double curve
22       22                                         Bumpy road
23       23                                      Slippery road
24       24                          Road narrows on the right
25       25                                          Road work
26       26                                    Traffic signals
27       27                                        Pedestrians
28       28                                  Children crossing
29       29                                  Bicycles crossing
30       30                                 Beware of ice/snow
31       31                              Wild animals crossing
32       32                End of all speed and passing limits
33       33                                   Turn right ahead
34       34                                    Turn left ahead
35       35                                         Ahead only
36       36                               Go straight or right
37       37                                Go straight or left
38       38                                         Keep right
39       39                                          Keep left
40       40                               Roundabout mandatory
41       41                                  End of no passing
42       42  End of no passing by vechiles over 3.5 metric ...
```

Figure 19: All 43 benchmark classes in GTSRB

In the figure 19 we can see that we have 43 classes starting from 0, Each number has a certain class for example Class 0 represents the sign name of speed limit up to 20 km/h it means if the Convolution neural network sees the sign of 20 km/hr then the output should be class 0The process will be same for the other classes.

6.2 Image Preprocessing

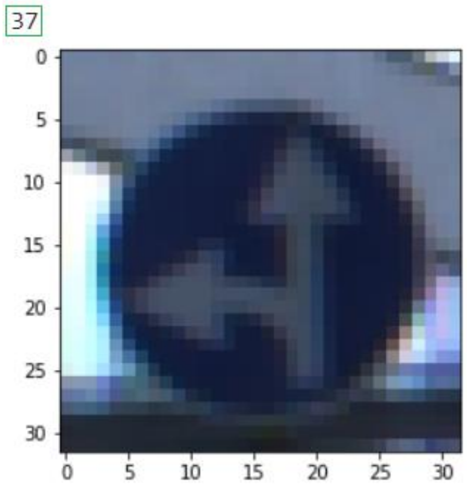
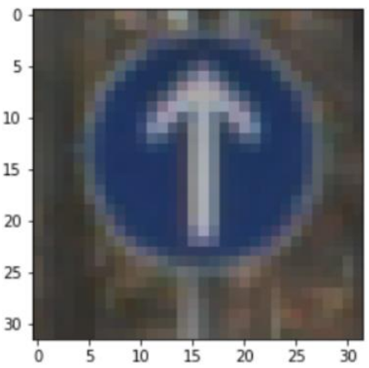


Figure 20: Random Image of go straight or left

The above Traffic sign is chosen randomly from the training set, the image has the shape of s 32 x 32 x 3 from class 37 (highlighted with green box) which indicates the sign 'go straight or Turn left' ,The output of class is consistent with respect to image.



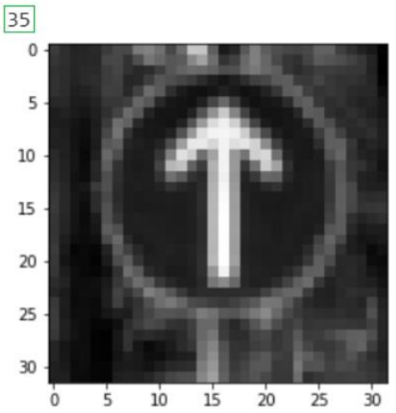


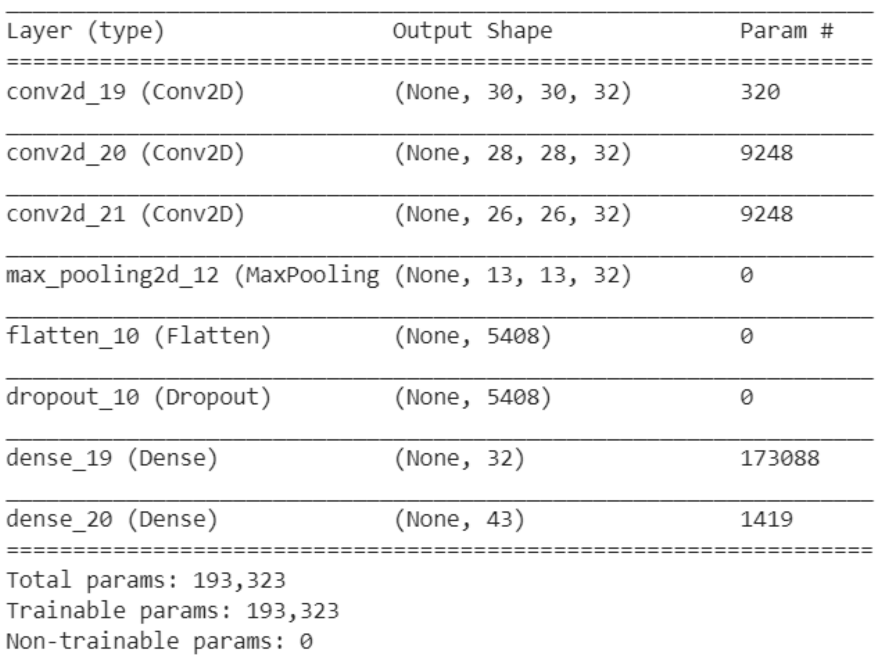
Figure 21 :Original Image

Figure 22 : Preprocessed Image

The next step is to convert our image from RGB to grey scale. The grey scale conversion is important for two main reasons the first being that one distinguishing between traffic signs. Colour is not a very significant feature to look for. The lighting in our image varies and many of the traffic signs of similar colours reinforcing that it is not a very relevant piece of information. The features of the traffic signs that really matter are the edges the curves the shape and side of it. The next reason is that when we convert an image from RGB to grayscale we reduce the depth of our image from three to one. This means that road network now requires fewer parameters as our input data will only have a depth of one channel. In the long run this means that our work will be much more efficient and will require less computing power to classify our data.

6.3 Model Training

We will begin coding this architecture by defining our leNet model function. We will define our neural model by setting it equal to sequential, now we can add layers to our neural network. The first layer which we are going to add is convolutional layer, because these convolutional layers consist of filters that recognize various features within the image. After adding the convolutional layer now we have to specify the shape of each input image which is fed in to the neural network is having a shape of 32 x 32 x 1. In figure 23The output of the layer demonstrates 30 x 30 x 32, which is quite sensible because our 32 x 32 image is now reduced to 30 x 30 with the depth of 32 ,since we are going to use 32 independent filters. Each filter adds extra depth to the convoluted image ,hence depth of 32 corresponds to 32 feature maps extracted by each kernel. The reason why each kernel map is scaled down to 30 x 30 is legitimate because of convolving 3 x 3 kernels with a stride of 1 we lose 2 pixel layer thickness at all borders of the image resulting in 30 x 30 for each feature map .There is no need to preserve the borders since looking at the traffic signs they seem to be centralized. And so the borders don’t really contain any significant features that are necessary for the neural network. The first convolutional layer network will have a total of 320 parameters((30(kernels) x 3 x3)+30(biased value)=320 Parameters) and it must adjust as each kernel has 9 adjustable parameters(3 x 3). The last argument for this convolutional layer is the activation function of the nodes which is going to be ReLu function. Now we are going to add another Convolutional layer to our LeNet model function from the figure 23 we can say that the output layer is now again reduced to 28 x 28 with a depth of 32,The second convolutional layer will have a total of 9248 parameters ((32(filters) *32(depth) *9(parameters/per filter))+32(biased)=9248 Parameters) rest of the arguments will be same as first convolutional layer .The Third layer in the model will be again a convolutional layer from the figure 23 we can see that the output layer is again reduced to 26 x 26 with a depth of 32,The parameters for the third convolutional layer is again   9248 parameters which is same as the second and the rest of the arguments are same as the previous one. Now we will add a Max pooling layer as per LeNet model architecture. The first argument of pooling element is defined by using pooling size 2 x 2. Pooling will scale down all the feature maps from the convolutional layer into a small abstracted generalized representation which ultimately helps to avoid overfitting and reduces the amount of parameters that are being adjusted. The next portion is going to be the Flatten function that we will use in to our network which will help us to flatten our data in order to format it properly so that it can be fed in to fully connected layer as one dimensional array. The next layer is the Dropout layer, The drop out layer recall takes an argument for fractioned rate which refers to the amount of nodes that drop out layer drops during each update with zero referring to a no inputs dropped and one refers to when all the input nodes are dropped. We will use arbitrary number 0.4.Now the next layer is Dense with output dimension =32, with activation Function = ‘ReLu’ which is kind of dimension of our hidden layer, Now for the Output dense layer the dimension is very important.The output dimension must be equal to number of classes we have, since we have 43 so the output dimension is 43 and the activation function is ‘softmax’.Now lastly we will compile our model using familiar optimizer adam with learning rate =’0.001’ and the model is compiled by defining the loss by setting it equal to ‘sparse_categorical_crossentropy’ as our network is categorising a multiclass data set, xince we are also interested in accuracy of our network so we can define metrics = ‘Accuracy’.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_19 (Conv2D)           (None, 30, 30, 32)        320       
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 28, 28, 32)        9248      
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 26, 26, 32)        9248      
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 13, 13, 32)        0         
_________________________________________________________________
flatten_10 (Flatten)         (None, 5408)              0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 5408)              0         
_________________________________________________________________
dense_19 (Dense)             (None, 32)                173088    
_________________________________________________________________
dense_20 (Dense)             (None, 43)                1419      
=================================================================
Total params: 193,323
Trainable params: 193,323
Non-trainable params: 0
```

Figure 23:LeNet model Architecture

6.4 Tuning the model

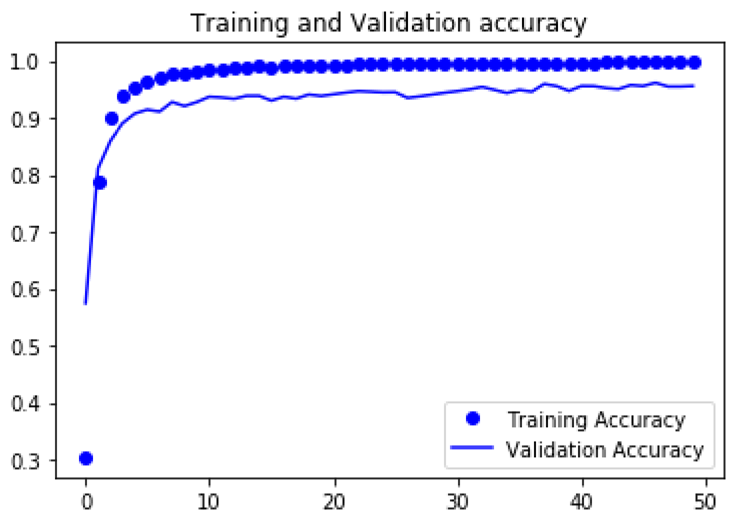


Figure 24:Training and Validation accuracy

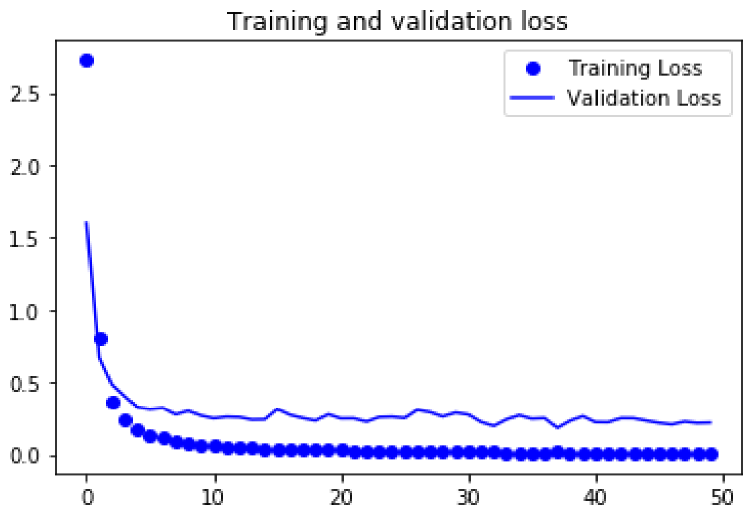


Figure 25: Training and validation Loss

From the figure 24 and 25 we can see that our model seems to be very effective, we have a very much smaller gap between our training loss and training accuracy as well our validation loss and Now we have a much smaller gap between our training loss and training accuracy as well as our validation loss and accuracy respectively, and this demonstrates consistency in our training and implies a better trained model. The model Training finished on a good note as we can see Training accuracy is quite good as it reached to 99.46% and the Validation accuracy reached to 93.49%. The test accuracy reached to 93.49%

Final Results

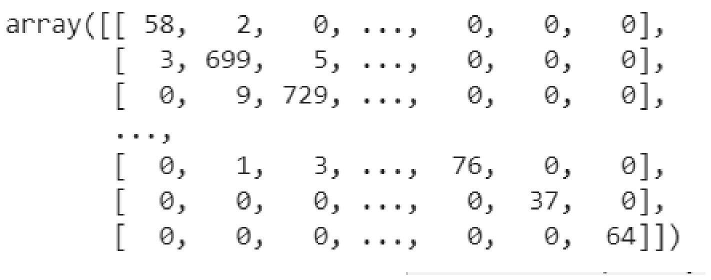
Training Accuracy- 99.46%

Validation Accuracy-93.49%

Test accuracy-94.43%

6.5 Confusion Matrix

The confusion matrix is use to evaluate our predicted class versus True class to see the number of samples classified correctly and from the figure 26 The diagonals element shows the number of elements classified correctly ,any other elements outside it are misclassified.



```
array([[ 58,   2,   0, ...,   0,   0,   0],
       [  3, 699,   5, ...,   0,   0,   0],
       [  0,   9, 729, ...,   0,   0,   0],
       ...,
       [  0,   1,   3, ...,  76,   0,   0],
       [  0,   0,   0, ...,   0,  37,   0],
       [  0,   0,   0, ...,   0,   0,  64]])
```

Figure 26: Confusion Matrix

6.6 Testing

For the testing part a loop of 12 random images was created to plot predictive class and true class. From the figure 27it can be said that the result was quite good as the margin of error for these 12 random images was zero. Each and every image was classified correctly.

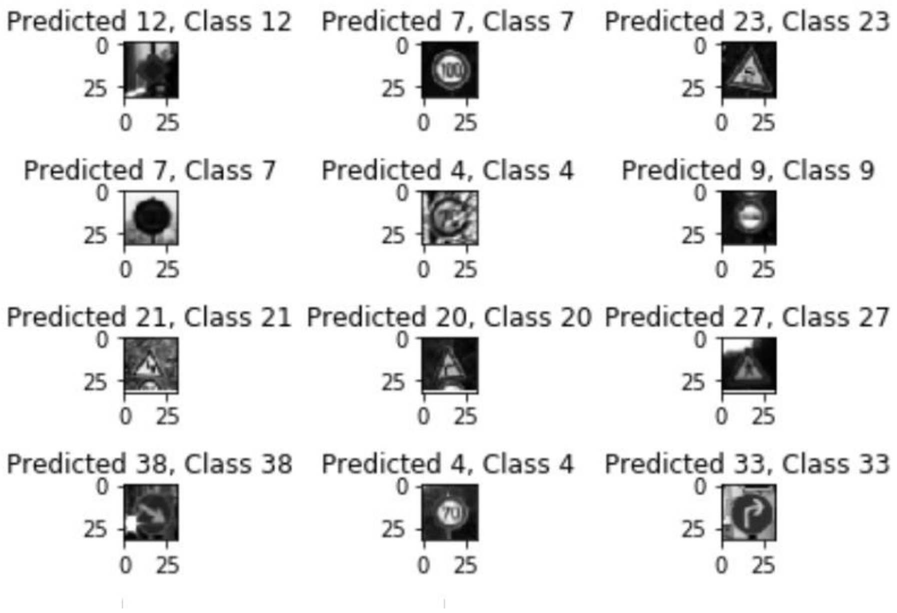


Figure 27 : Predicted class v real class

	Predicted Class	True Class
1.	12-Priority road	12-Priority road
2.	7-Speed limit (100 km/h)	7-Speed limit (100 km/h)
3.	23-Slippery road	23-Slippery road
4.	7- Speed limit (100 km/h)	7- Speed limit (100 km/h)
5.	4-Speed limit (70km/h)	4-Speed limit (70km/h)
6.	9-No passing	9-No passing
7.	21-Double Curve	21-Double Curve
8.	20-Dangerous Curve to right	20-Dangerous Curve to right
9.	27-Pedestrians	27-Pedestrians
10.	38-Keep Right	38-Keep Right
11.	4--Speed limit (70km/h)	4--Speed limit (70km/h)
12	33_Turn Right ahead	33_Turn Right ahead

Table 1 Output Result of Predicted class v/s True Class

6.7 Conclusion

I successfully performed the experiment of Convolutional Neural Network to the Traffic Sign Recognition task. Opensource libraries like keras and Tensor flow were used.The model which I used was LeNet Architecture .While performing the Experiment Several times at last I got the validation accuracy of 99.46% .Many deep learning system are using the CNN architectures like GoogleNet ,ResNet but the computational cost for these networks are pretty high. The interesting part playing with the architecture as after several tries I got the 100% result in detecting the images through my LeNet model.

7.0 Lane Detection Result

7.1 Canny Edge Detection Technique:

Figure 28 is the gradient image which clearly traces an outline of the edges that correspond to the most sharp changes in intensity, gradients that betters the high threshold are known as bright pixels, recognising adjacent pixels in the image with the most fast changes in brightness. Small changes in brightness are not traced.

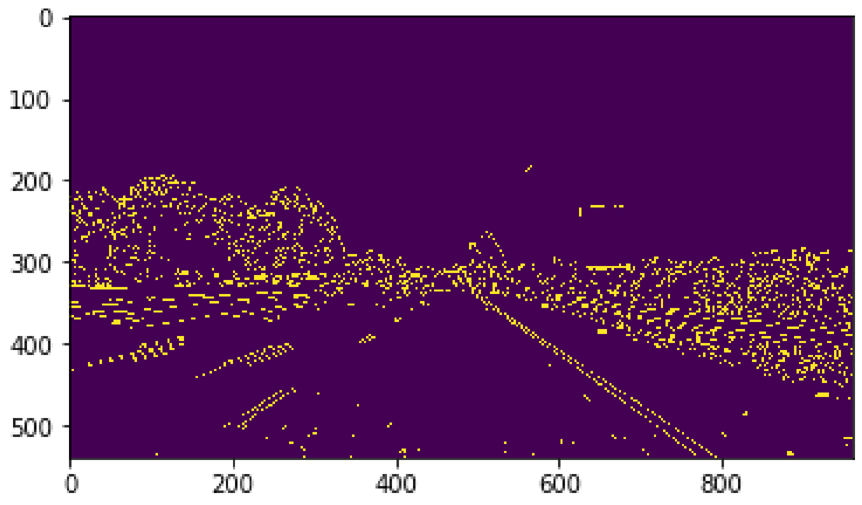


Figure 28 : Canny image

7.2 Region of Interest:

The image which contains the road lanes with its chosen dimension is marked as our region of interest. The rest of the area which would be an array of zeros, now the next task is to fill the region of interest in the mask with the depth of 255 so that our region of interest are grey, then we do bitwise AND operation and the mask which will result in our final region of interest

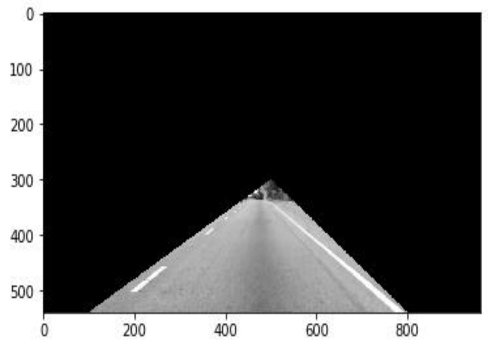


Figure 29:ROI

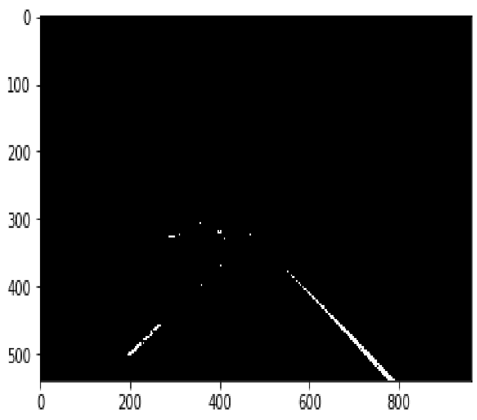


Figure 30: image after doing a bitwise operation with the canny image and the mask

7.3 Hough Transform:

Now we make use of Hough transform technique that will detect straight lines in the image and thus identify the lane lines. We know that a straight line is represented by the below equation:
y= mx + b
And the slope of the line is simply a rise over run. If the y intercept and slope is given then the line can be plotted in the Hough Space as a single dot. There are many possible lines that can pass through this dot each line with different values for M and B. There are many possible lines that can cross each point individually, each line with different slope and y intercept values. However there is one line that is consistent with both points. We can identify that by looking at the point of intersection enough space because that point of intersection in Hough Space and that point of intersection represents the M and B values of a line steady with crossing both the points. Now in order to identify the lines, we will first split our Hough space into a grid. Each bin inside the grid corresponding to the slope and y intercept value of the line. For every point of intersection in a Hough Space bin we're going to cast a vote inside of the bin that it belongs to. The bin with maximum number of votes is going to be our line. But as we know that the slope of a vertical line is infinity. So to express vertical lines, we will use polar coordinates instead of cartesian coordinates.

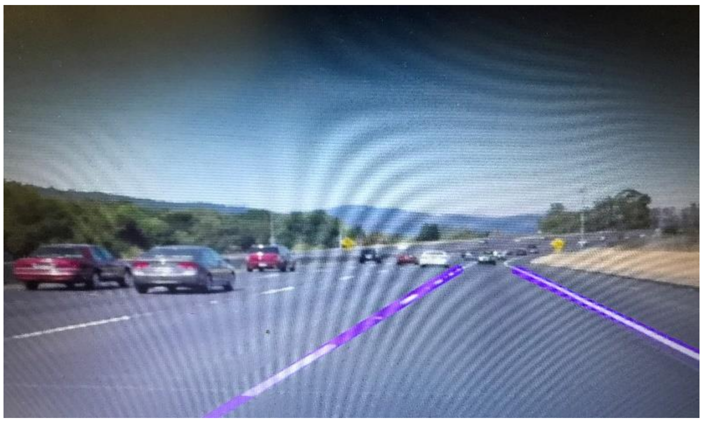


Figure 31: detection of lane (detected in Video)

7.4 Conclusion

In the experiment , we had used OpenCV library and its functions such as the Canny Function through which we accomplished edge detection. Then we processed a mask of zero intensity and mapped our region of interest by using the bitwise operation. Then Hough Transform technique was used which detected the straight lines in the video and identified the lane lines.

8.0 Udacity’s Car Simulation Result

8.1 Data adjustment

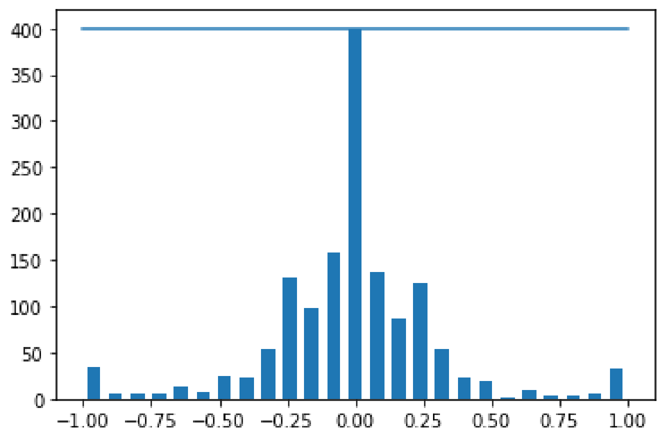


Figure 32: Histogram of Steering angles

Figure 32 shows Histogram steering angles from -1 to 1 centered around a steering angle of zero. This vertical axis specifying the values of our histogram, the frequency of each steering angle during the simulation. Plotting the histogram of training data shows us that the data we obtained from driving in Udacity track one has more zeros angles because of the nature of the track as in most of the time simulator was drove down the middle of a straight track, while the left and right steering angle seemed to be pretty balanced.

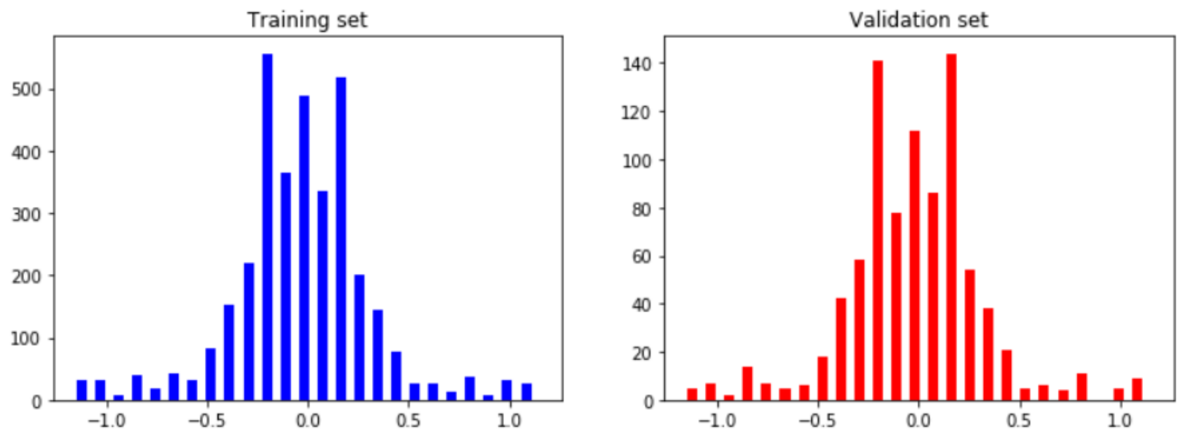


Figure 33: Training set and Validation set

From the figure 33 It's clear that the data isn't exactly similar but both follow a general trend such that both right and left steering angles are pretty balanced. The data is still somewhat biased towards the center. Meaning that splitting our data was successful

8.2 Image Preprocessing

From the original image we can see that there are a lot of features that are not very important to focus on. For example the top of our image is almost entirely just scenery consisting of trees and mountains. While the very bottom of our image is just the HUD of our own Simulator these features are not relevant for helping our simulator to determine the steering angle as there is no relationship between the scenery surrounding the road and the steering angle of our simulator. For this reason the image was cropped it was also cropped tp remove the noise and other unnecessary features. Our image is essentially just a three dimensional array which contains a height width and a channel index as we were only concerned with cropping our image height. From the axis of our image we can see that the top of our image has a value of 0. While the bottom of our image has a value of 160 at a value of approximately 60 we can see that we can crop out all the unnecessary features on top of the image and on the lower end of our image. Next step was to convert the image from RGB to YUV where Y represents the brightness and UV component represents chromium which add colours to the image. Next method was to use the gaussian blur function to smoothen the image and to remove noise within the image, by removing unnecessary features will allow our model to focus on the more important features of the image such as the lane lines and the borders.

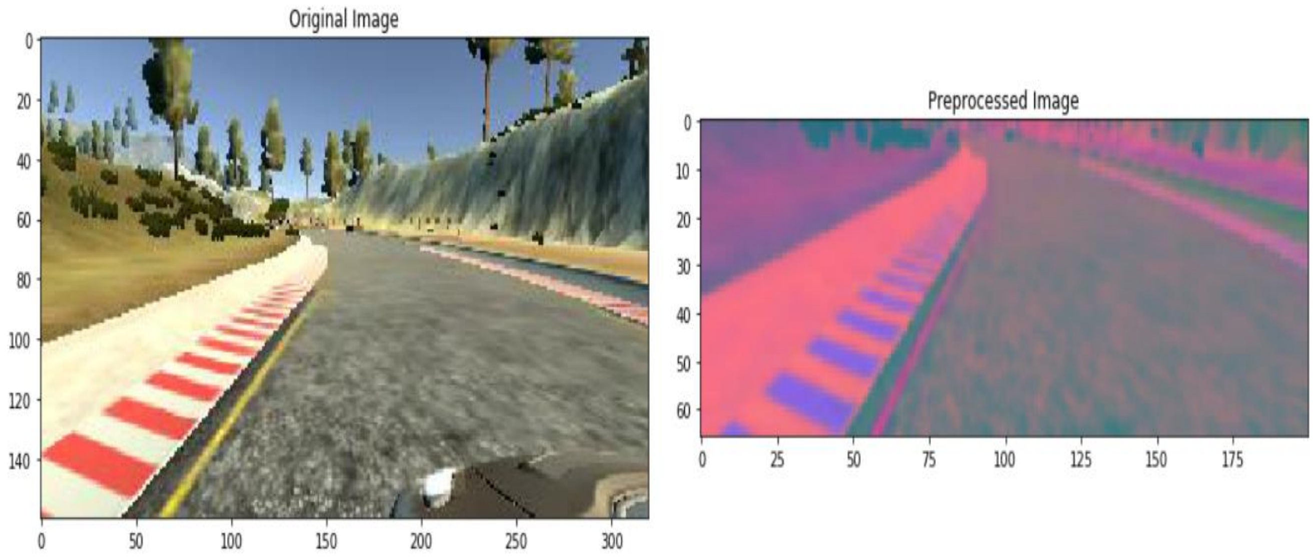
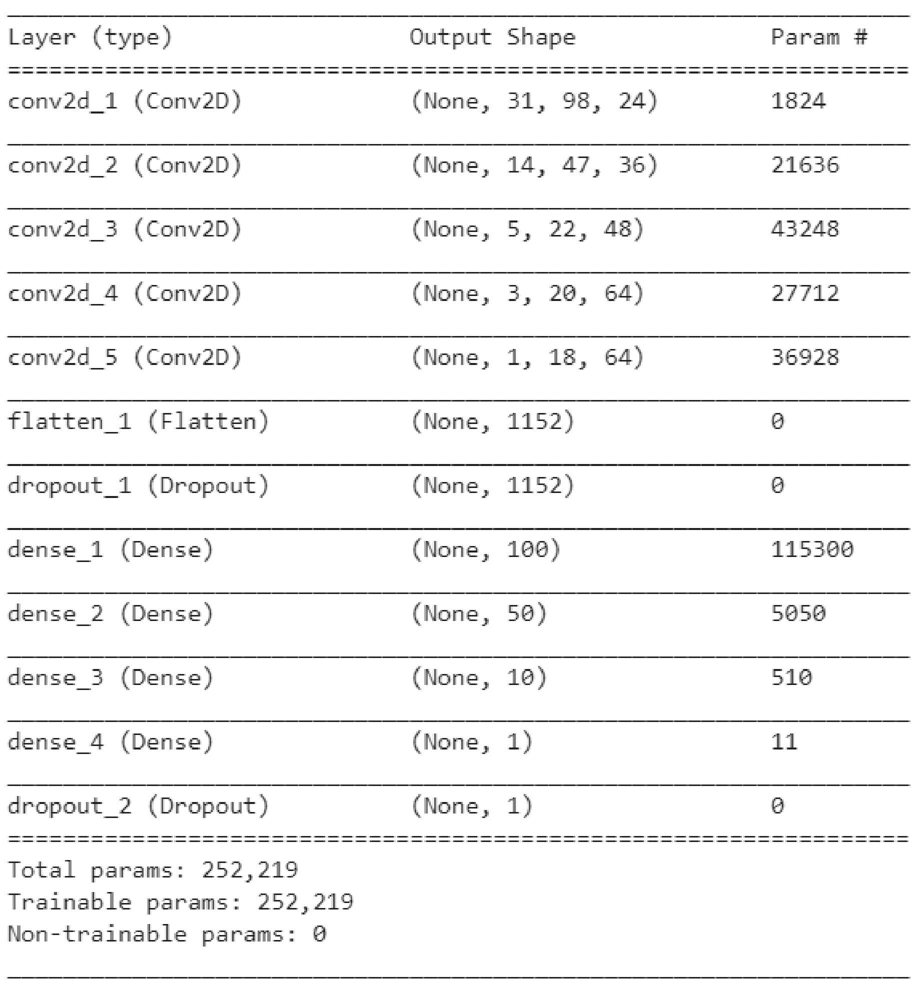


Figure 34: Original image v/s Preprocessed image

8.3 Nvidia Model



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
_________________________________________________________________
dropout_2 (Dropout)          (None, 1)                 0
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
_________________________________________________________________
```

Figure 35: Nvidia model architecture

In previous Section LeNet model was used in classifying our traffic sign dataset However with behavioral cloning our data set is more complex than any of the other datasets that we had dealt with before. in previous section he traffics sign images were 32 by 32. However now we are dealing with images that have a larger dimension our current dataset only has 1010 images to train with. while our previous codes dealt with classification problems our behavioral cloning code simply has to return the appropriate steering angle which is a regression type example. Because of the complexity we need a much more capable model. From the figure 35 we can see that we have the five convolutional layers that we added at the beginning of our model along with the appropriate number of filters at each[48] layer.As expected these layers are followed by a drop out layer and now the next layer is flatten layer which flattens the data from our last convolutional layer to then use it in the dense layer that follows. And finally we can see our final four dense layers including our last output layer which contains only single node There is also a drop out layer between our first dance layer and the second dense layer to prevent against overfitting and the total trainable parameters of our model results in 252,219

8.4 Data Augmentation

Designing our own generator provides us with added flexibility as it allows us to add or remove any of the augmentation techniques that we require. This can help us to customize our own data augmentation to very specific data sets or to very specific applications. By designing our own data generator we will gain some transparency with regards to the process of data augmentation which will help us better understand it’s useful technique for improving the quality of the training process

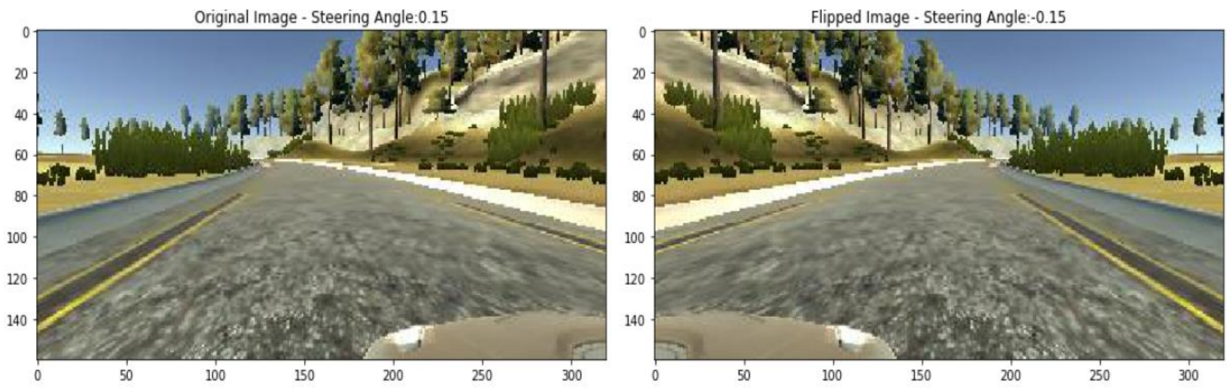

Original Image - Steering Angle:0.15
Flipped Image - Steering Angle:-0.15

Figure 36: augmented flipped image from original image

Figure 36 shows a image of right turn at angle of 0. 15 degrees which is flipped in to -0.15 degrees,this is very good for dataset, now we have data that accounts the turn on both sides.

8.5 Batch Generator

The generator allows us to create small batches of images at a time only when the generator is actually called. This is much more memory efficient as data is only used when it's required rather than being stored in memory. In our previous section traffic sign code has images that were 32 x 32 which is very small in comparison to our images that are 320 x 160 these memory issues become even more complex with larger data sets which .makes the use of an image generator very important

- Small datasets:

  fit()

  predict()

- Larger datasets

Fit_generator()

Predict_generator()

This function is not like other typical functions normal functions begin executing the functions as first line and continue until they hit some kind of return statement at the end of the function. At this point any of the local values inside the function are re-nitialized and any new call to the unction recreates everything these functions are referred to as subroutines. This generator uses the yield keyword.

```
def generator_name()
#statement
yield something
```

Each time the yield statement is executed the function generates a new rule.

```
def yrange(n):
      i=0
       while i<n:
        yield i
        i+=1
```

8.6 Fit generator

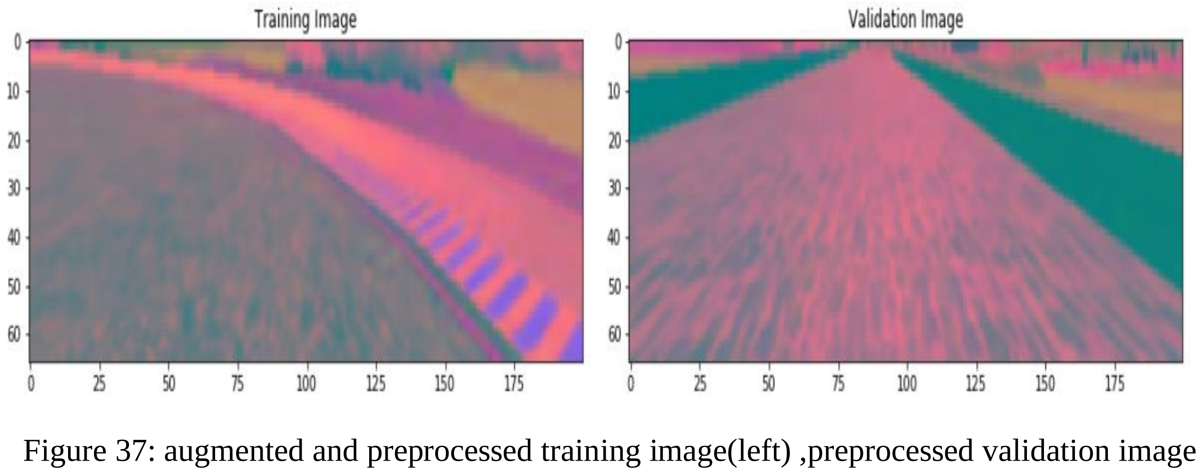


Figure 37: augmented and preprocessed training image(left) ,preprocessed validation image without augmentation(right)

We can see that while our training images have more added variety due to the augmentations being applied. The validation images remain consistent as they have not been augmented.This reinforces the idea that augmenting our training sets helps our model generalize to new data. As we are dealing

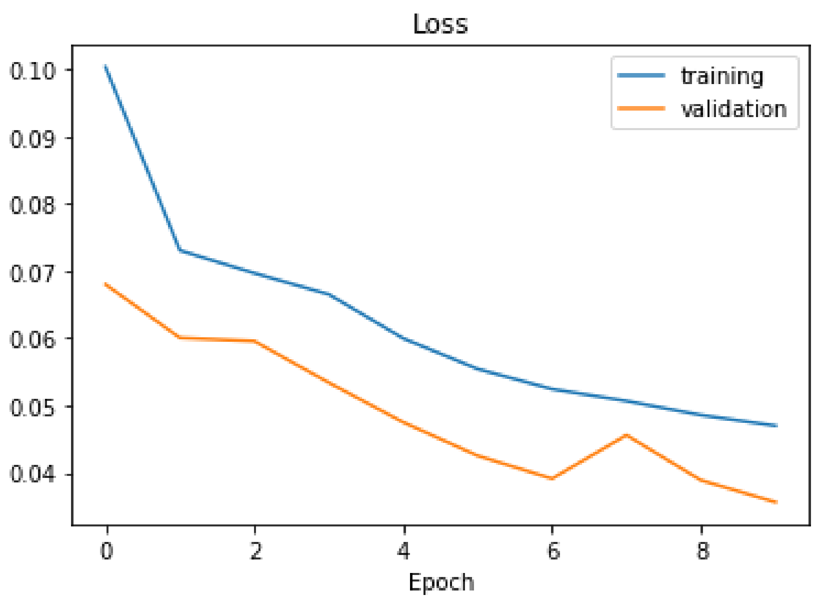


Figure 38-training and validation loss

Now from figure 38 we can say that the model is not overfitting and the loss has been minimised so we used the above model to validate in to our system,

8.7 Conclusion

After collecting data by driving manually for 3 laps , then we validated that data in to the system by different Techniques. we started from the Data adjustment where we adjusted the data by cropping the graph to 400, so that data should be distributed conveniently in to left and right. Then we went through the image preprocessing where we fed the system only the required data rest of the useless data like HUD of the Car and Trees were Cropped for the betterment of the model so that model can define the road easily, Then we went on to make the Nvidia model by adding several convolutional layers, dropouts, flattens after that the data augmentation technique was used where the data was Zoomed ,pan, and flipped to provide flexibility to our model. Lastly fit generator and batch generator were used to over come model fitting, once the model was prepared it was successfully validated in to the udacity real car engine.

9.0 References & Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in Advances in neural information processing systems, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, ArXiv preprint arXiv:1409.1556, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, ArXiv preprint arXiv:1512.03385, 2015.

[4] D. Liden, „What Is a Driverless Car?,” 2017. [Online]. Available: http://www.wisegeek.com/what-is-a-driverless-car.htm.

[5] C. Mercer, „12 Companies Making Driverless Cars You Should Know About.,” 2017. [Online]. Available: http://www.techworld.com/picture-gallery/data/-companies-working-on-driverless-cars-3641537/.

[6] L. Blain, „Self-driving vehicles: What are the six levels of autonomy?,” 2017. [Online]. Available: http://newatlas.com/sae-autonomous-levels-definition-self-driving/49947.

[7] https://roboticsandautomationnews.com/2017/07/01/adas-features-of-advanced-driver-assistance-systems/13194/

[8] https://www.businesswire.com/news/home/20170912005580/en/Mobileye-Munich-Announce-Collaboration-Reduce-Automotive-Collisions

[8] C. Mercer, „12 Companies Making Driverless Cars You Should Know About.,” 2017. [Online]. Available: http://www.techworld.com/picture-gallery/data/-companies-working-on-driverless-cars-3641537/.

[10] https://www.i-micronews.com/how-lidar-is-getting-ready-for-the-automotive-mass-market-an-interview-with-velodyne/

[11] https://www.slashgear.com/googles-self-driving-car-could-lose-its-hat-with-new-laser-tech-06332372/

[12] https://www.aitrends.com/ai-insider/crossing-the-rubicon-and-ai-self-driving-cars/

[13] Guangqian Lu, A Lane Detection, Tracking and Recognition System for Smart Vehicles,2015

[14]https://arstechnica.com/cars/2018/01/driving-around-without-a-driver-lidar-technology-explained/

[15] https://www.aitrends.com/ai-insider/crossing-the-rubicon-and-ai-self-driving-cars/

[16] Simon Haykin. *Neural Networks. A comprehensive foundation*. Hamilton, Ontario, Canada: Pearson Education, 1999.

[17] Model of an artificial, neuron.https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm

[18] Binary step function https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/

[19] Umberto Michelucci,Applied Deep learning:A case based approach to understanding deep neural networks, Apress,2018

[20] A. Krizhevsky, I. Sutskever, and G. Hinton. "Imagenet classification with deep convolutional neural networks". NIPS, 2012.

[21]multilayer neural networks (only image is taken), https://fr.m.wikipedia.org/wiki/Fichier:MultiLayerNeuralNetworkBigger_english.png

[22] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, no. 8, pp. 1798–1828, 2013.

[23] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning", Book in preparation for MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org.

[24] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115–133, 1943.

[25] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.", Psychological review, vol. 65, no. 6, p. 386, 1958.

[26] (PhD thesis, 1974), although several other researchers individually re-discovered it in the following years

[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", Neural networks, vol. 2, no. 5, pp. 359–366, 1989.

[28] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks", in Advances in Neural Information Processing Systems, 2011, pp. 666–674.

[29] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", Biological cybernetics, vol. 36, no. 4, pp. 193–202, 1980.

[30] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex", The Journal of physiology, vol. 160, no. 1, pp. 106–154, 1962.

[31] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network", in Advances in neural information processing systems, Citeseer, 1990.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets”, Neural computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[34] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, “Flexible, high performance convolutional neural networks for image classification”, in IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, 2011, p. 1237.

[35] [35] Adventures in machine learning, Convolutional Neural Networks Tutorial in TensorFlow;Cited: Nov 20 2017. Website. Available from: http://adventuresinmachinelearning:com/ convolutional-neural-networks-tutorial-tensorflow/.
.

[36] Jianxin Wu. *Introduction to Convolutional Neural Networks*. 2017. url: https:// pdfs.semanticscholar.org/450c/a19932fcef1ca6d0442cbf52fec38fb9d1e5.pdf.

[37] Http://cs231n.stanford.edu/, “Convolutional Neural Networks (CNNs / ConvNets),” 2019 /.
[38] P. Pošík and P. Pošík, “CZECH TECHNICAL UNIVERSITY IN PRAGUE Faculty of Electrical Engineering Department of Cybernetics Neural Networks. Introduction and Rehearsal,” tech. rep.

[39] [23] F. Chollet *et al.*, “Keras.” https://keras.io, 2015.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

url: http://www.deeplearningbook.org

[41] [24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S.Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur,J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. Software available from tensorflow.org.

[42] https://medium.com/pursuitnotes/day-38-deep-learning-3-ann-2-c3c999f6cef9

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014.

[44] Colab.research.google.com, “Welcome to Colaboratory!.”

[45] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine le

[46] Velickovic P. Cambridge Spark, Deep learning for complete beginners: convolutional neural networks with keras; Cited: Nov 7 2017. Website. Available from: https://cambridgespark:com/content/tutorials/convolutional-neuralnetworks- with-keras/index:html.

[47] Lihua Wen and Kang-Hyun Jo, "Traffic Sign Recognition and Classification with Modified Residual Networks",2017

[48] End to End Deep Learning For Self Driving cars, https://devblogs.nvidia.com/deep-learning-self-driving-cars/

[49] https://github.com/llSourcell/How_to_simulate_a_self_driving_car

[50]https://github.com/ndrplz/self-driving-car/tree/master/project_1_lane_finding_basic

[51] https://github.com/ndrplz/self-driving-car