Ahsanullah University of Science & Technology

Department of Computer Science and Engineering

A Study on Traffic Lights and Car Brake-Light Detection and Classification For Autonomous Car

Submitted By:

Swopnil Dewan	**14.01.04.006**
Alauddin Al Azad	**14.01.04.067**
Nafis Islam	**14.02.04.014**
Hafizul Islam Himel	**14.02.04.024**

Supervised By:

Mr. Md. Wasi Ul Kabir
Assistant Professor, Dept. of CSE,
Ahsanullah University of Science & Technology

Contents

1 Abstract **4**

2 Introduction & Motivation **4**

3 Problem Description **5**
3.1 Problem Solving Pipeline 5

4 Deep Neural Network **6**
4.1 Biological Motivation and Connections 7
4.2 Mathematical Background of Neural Network 8
4.3 Notation 8
4.4 Forward Propagation Equations 9
4.5 Backward Propagation Equations 9
4.6 Building Block of Deep Neural Network 10
4.7 Dimensions of the Parameters 11

5 Deep Neural Network Training **11**
5.1 Training data 11
5.2 Choose appropriate activation functions 11
5.2.1 Rectified Linear Units(ReLu) 12
5.2.2 Softmax 12
5.3 Number of Hidden Units and Layers 12
5.4 Weight Initialization 12
5.5 Learning Rates 12

6 Learning Method **13**
6.1 Gradient Descent 13
6.2 Stochastic Gradient Descent 13

7 One Hot Encoding **14**

8 Dataset Analysis **14**

9 Our Own Data set **15**

10 Classification **16**

11 Results & Analysis **17**

12 Future Works **18**

13 Conclusion **18**

List of Figures

1 Traffic Light and Car-brake light . . . 5
2 Deep Neural Network . . . 6
3 Biological Motivation and Connections . . . 7
4 Two layer Neural Network . . . 8
5 Mathematical figure of Two layer Neural Network . . . 8
6 Building Block of Deep Neural Network . . . 10
7 Deep Neural Network Pipeline . . . 11
8 Gradient Descent . . . 13
9 One Hot Encoding . . . 14
10 Sample pictures of training data set . . . 14
11 Our Own Data set . . . 15
12 Labeled brake light images of Bosche Dataset . . . 15
13 Classification Pipeline . . . 16
14 Classification Pipeline . . . 17

1 Abstract

An intensely researched problem which has led to the appearance of many driver assistance systems is the Autonomous vehicle or self-driving car. Many challenges are faced by the Metropolitan areas which requires more refined algorithms in various areas ranging from perception over behavioral planning to accident prevention systems. The detection and classification of traffic lights in real-time is an important part. Traffic lights present a challenging problem due to their small size and high ambiguity with other objects present in the urban environment, such as lamps, decorations, and reflections. Traffic Light Detection and classifying their state at intersections plays the main role in driver assistance systems or autonomous vehicles. Right now, there are no systems that can accurately understand traffic lights in real-time, without map-based information, and in sufficient distances needed for smooth urban driving. Autonomous cars have built in camera inside them. Using video stream taken from the camera we can apply traffic light detection and also know the current state of the lights. Brake-light detection for prevention of rear-end collision is also a crucial part for autonomous driving.

2 Introduction & Motivation

An autonomous car (also known as a driverless car, self-driving car, robotic car) and unmanned ground vehicle is a vehicle that is capable of sensing its environment and navigating without human input. Autonomous cars use a variety of techniques to detect their surroundings, such as radar, laser light, GPS, odometry and computer vision [1]. Autonomous cars must have control systems that are capable of analyzing traffic state on the road. Traffic light detection and recognition is essential for autonomous driving in urban environments. A camera based algorithm for real-time robust traffic light detection and recognition was proposed, and especially designed for autonomous vehicles. Although the current reliable traffic light recognition algorithms operate well under way, most of them are mainly designed for detection at a fixed position and the effect on autonomous vehicles under real-world conditions is still limited! [2]. In the context of autonomous vehicles, a basic requirement is being able to follow vehicles safely and take actions properly to prevent rear-end collisions and accidents. Rear-lights signal recognition is important for a driver in understanding the behavior and intention of the vehicle ahead.

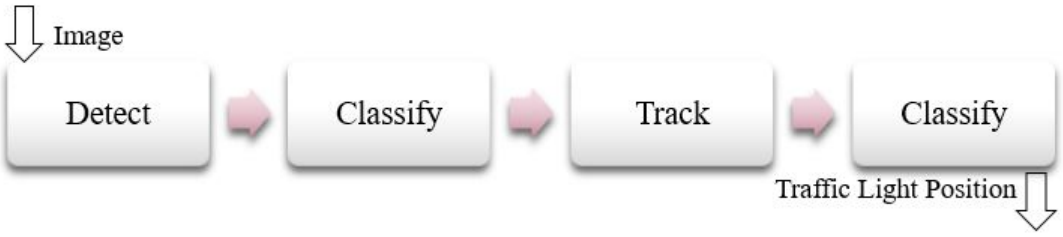
Autonomous car is one of the most debated technology in the automobile industry right now and many companies are developing autonomous driving features. The motivation of this technology is to maximize car safety and efficiency. According to the World Health Organization, more than 1 million people lose their lives on the road due to car accidents, and C2ES (Center for Climate and Energy Solutions) states that about 60% of total energy consumed by transportation is from automobiles. There have been attempts to solve these problems, but none of the solutions have been particularly effective. However, self-driving technology has potential to change all these problems. If self-driving cars can be driven safer and much more efficiently, it could save valuable lives and preserve the environment [3]. As we proposed a model to implement detection of traffic lights and classify it's states, we implemented only classification part with deep neural network so far.

3 Problem Description

An autonomous car is a vehicle that can guide itself without human conduction. This kind of vehicle has become a concrete reality and may pave the way for future systems where computers take over the art of driving. An autonomous car is also known as a driverless car, robot car, self-driving car or autonomous vehicle [**?**]. Autonomous cars use various kinds of technologies. They can be built with GPS sensing knowledge to help with navigation. They may use sensors and other equipment to avoid collisions. To avoid collision sensors are not enough in fact they are not so robust to maintain the traffic rules . For this reason we use modern technology called Computer Vision. So, our proposed model was Traffic light and Car Brake light detection and classification to maintain the traffic rules with better accuracy. The main concept of our problem is, an autonomous car will take frames from video stream while driving on the road and detect traffic lights, car brake lights and classify them.

3.1 Problem Solving Pipeline

We divided our whole problem into several segments. So, the pipeline of our proposed model is shown in the above figure:



Some of the images of Traffic light & Car-brake light.

Figure 1: Traffic Light and Car-brake light

4 Deep Neural Network

Deep feedforward networks, also called **feedforward neural networks**, or **multilayer perceptrons (MLPs)**, are the quintessential deep learning models.The goal of a feedforward network is to approximate some function f. For example,for a classier, **y = f(x)** maps an input **x** to a category **y**. A feedforward network denes a mapping **y=f(x;θ)** and learns the value of the parameters that result in the best function approximation. These models are called feedforward because information ows through the function being evaluated from **x**, through the intermediate computations used to dene **f**, and nally to the output **y**. There are no feedback connections in which outputs of the model are fed back into itself [3].

Feedforward neural networks are called networks because they are typically represented by composing together many dierent functions. The model is associated with a directed acyclic graph describing how the functions are composed together. For example, we might have three functions $f^{(1)}$,$f^{(2)}$, and $f^{(3)}$ connected in a chain, to form f(x) = $f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These chain structures are the most commonly used structures of neural networks. In this case,$f^{(1)}$is called the rst layer of the network,$f^(2)$is called the second layer, and so on [3].

The overall length of the chain gives the depth of the model. The name "deep learning"arose from this terminology. The nal layer of a feedforward network is called the output layer. During neural network training, we drive f(x) to match f^*(x).The training data provides us with noisy, approximate examples of f(x) evaluated dierent training points. Each example **x** is accompanied by a label **y** f(x). The training examples specify directly what the output layer must do at each point **x**; it must produce a value that is close to **y**. The behavior of the other layers are not directly specied by the training data. The learning algorithm must decide how to use those layers to produce the desired output, but the training data do not say what each individual layer should do. Instead, the learning algorithm must decide how to use these layers to best implement an approximation of f^*. Because the training data does not show the desired output for each of these layers, they are called **hidden layers** [3].

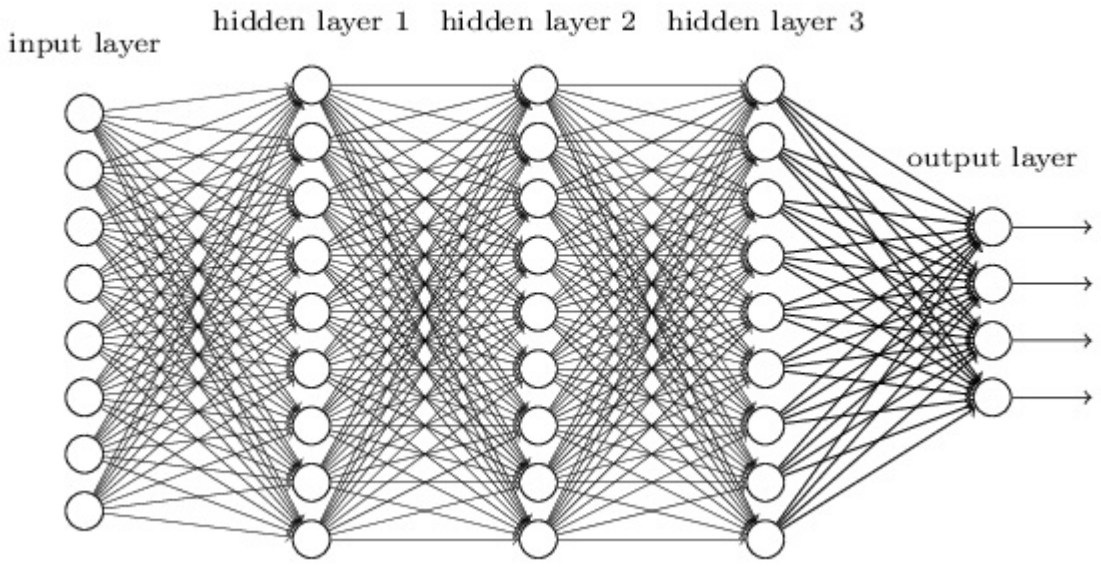


Figure 2: Deep Neural Network

4.1 Biological Motivation and Connections

The basic computational unit of the brain is a neuron. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} - 10^{15} synapses. The diagram below shows a cartoon drawing of a biological neuron (left) and a common mathematical model (right). Each neuron receives input signals from its dendrites and produces output signals along its (single) axon. The axon eventually branches out and connects via synapses to dendrites of other neurons. In the computational model of a neuron, the signals that travel along the axons (e.g. $\mathbf{x}$) interact multiplicatively (e.g. $\mathbf{w}$*$\mathbf{x}$) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g. $\mathbf{w}$).

The idea is that the synaptic strengths (the weights $\mathbf{w}$) are learnable and control the strength of influence (and its direction: excitory (positive weight) or inhibitory (negative weight)) of one neuron on another. In the basic model, the dendrites carry the signal to the cell body where they all get summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that the precise timings of the spikes do not matter, and that only the frequency of the firing communicates information. Based on this rate code interpretation, we model the firing rate of the neuron with an **activation function f**, which represents the frequency of the spikes along the axon. Historically, a common choice of activation function is the **sigmoid** function σ, since it takes a real-valued input (the signal strength after the sum) and squashes it to range between 0 and 1 [4].

Nueral networks are called neural because they are loosely inspired by neuroscience. Each hidden layer of the network is typically vector valued. The dimensionality of these hidden layers determines the width of the model. Each element of the vector may be interpreted as playing a role analogous to a neuron.Rather than thinking of the layer as representing a single vector-to-vector function,we can also think of the layer as consisting of many units that act in parallel,each representing a vector-to-scalar function. Each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value. The idea of using many layers of vector-valued representations is drawn from neuroscience. The choice of the functions $f^{(i)}(\mathrm{x})$ used to compute these representations is also loosely guided by neuroscientic observations about the functions that biological neurons compute. Modern neural network research,however, is guided by many mathematical and engineering disciplines, and the goal of neural networks is not to perfectly model the brain. It is best to think of feedforward networks as function approximation machines that are designed to achieve statistical generalization, occasionally drawing some insights from what we know about the brain, rather than as models of brain function [3].

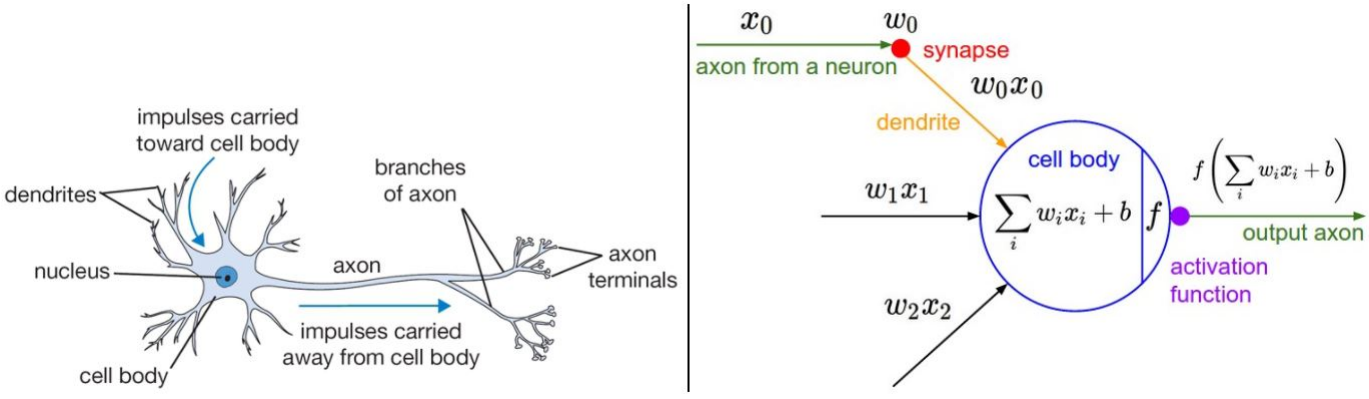


Figure 3: Biological Motivation and Connections

4.2 Mathematical Background of Neural Network

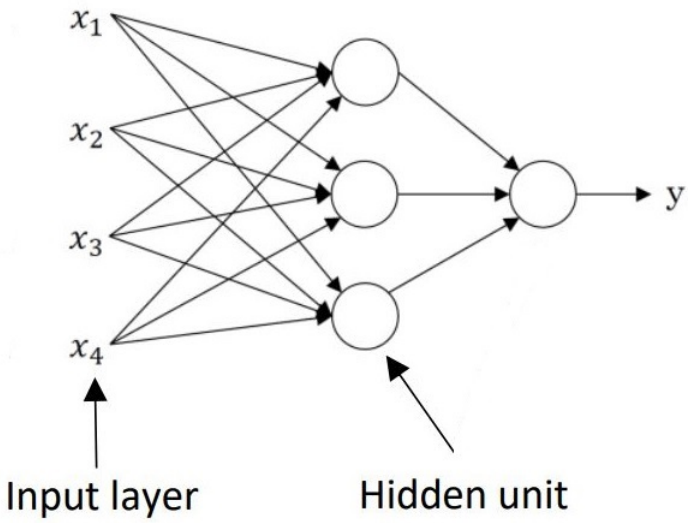


Figure 4: Two layer Neural Network

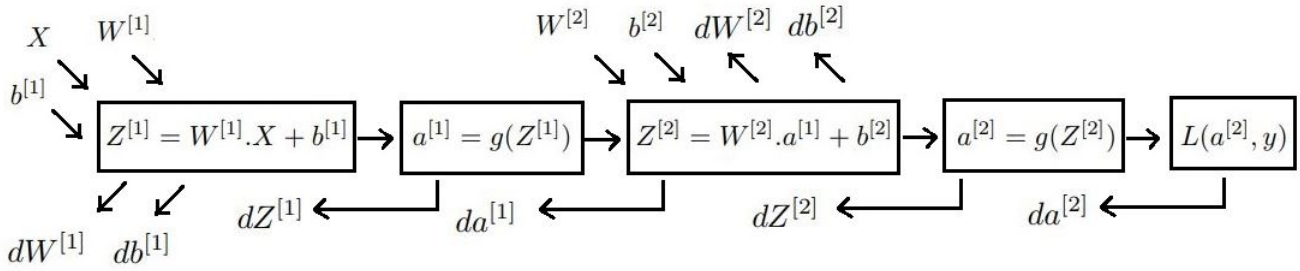


Figure 5: Mathematical figure of Two layer Neural Network

4.3 Notation

- $\mathrm{X} \in \mathbb{R}^{\mathrm{n_x} \times \mathrm{m}}$ is the input matrix
- $x^{(i)} \in \mathbb{R}^{\mathrm{n_x}}$ is the i^{th} example represented as a column vector
- $\mathrm{Y} \in \mathbb{R}^{\mathrm{n_y} \times \mathrm{m}}$ is the label matrix
- $y^{(i)} \in \mathbb{R}^{\mathrm{n_y}}$ is the output label for the i^{th} example
- $W^{[l]} \in \mathbb{R}^{\text{number of units in next layer} \times \text{number of units in the previous layer}}$ is the weight matrix, superscript [l] indicates the layer
- $b^{[l]} \in \mathbb{R}^{\text{number of units in next layer}}$ is the bias vector in the l^{th} layer
- $a^{[l]} = g^{[l]}(W_x x^{(i)})$ where $g^{[l]}$ denotes the l^{th} layer activation function
- $Z^{[l]} = g^{[l]}(W_x x^{(i)})$
- $\mathcal{L}(\hat{y},\mathrm{y})$ cost function of the output layer [5]

4.4 Forward Propagation Equations

$$Z^{[l]} = W^{[l]}.a^{[l-1]} + b^{[l]}$$
$$Z^{[l]} = g^{[l]}(Z^{[l]})$$

where, l = layer number

4.5 Backward Propagation Equations

$\frac{\partial}{\partial a^{[L]}}\mathcal{L}(\hat{y}, y)$ = Partial differentiation of the output layer cost function with respect to output layer's activation

$$\begin{aligned}\frac{\partial}{\partial a^{[L]}}\mathcal{L}(\hat{y}, y) &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l+1]}} \cdot \frac{\partial Z^{[l+1]}}{\partial a^{[l]}} \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l+1]}} \cdot \frac{\partial}{\partial a^{[l]}}(W^{[l+1]} \cdot a^{[l]} + b^{[l+1]}) \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l+1]}} \cdot (W^{[l+1]})\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial Z^{[l]}}\mathcal{L}(\hat{y}, y) &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[l]}} \cdot \frac{\partial a^{[l]}}{\partial Z^{[l]}} \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[l]}} \cdot \frac{\partial}{\partial Z^{[l]}} g^{[l]}(Z^{[l]}) \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[l]}} \cdot g^{'[l]}(Z^{[l]})\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial W^{[l]}}\mathcal{L}(\hat{y}, y) &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l]}} \cdot \frac{\partial Z^{[l]}}{\partial W^{[l]}} \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l]}} \cdot \frac{\partial}{\partial W^{[l]}}(W^{[l]} \cdot a^{[l-1]} + b^{[l]}) \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[l]}} \cdot a^{[l-1]}\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial b^{[l]}}\mathcal{L}(\hat{y}, y) &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l]}} \cdot \frac{\partial Z^{[l]}}{\partial b^{[l]}} \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l]}} \cdot \frac{\partial}{\partial b^{[l]}}(W^{[l]} \cdot a^{[l-1]} + b^{[l]}) \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l]}} \cdot 1 \\ &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial Z^{[l]}}\end{aligned}$$

$$
\begin{aligned}
where, L &= \text{the last layer or the output layer} \\
l &= \text{layer number} \\
\hat{y} &= \text{Predicted class} \\
y &= \text{True class}
\end{aligned}
$$

4.6 Building Block of Deep Neural Network

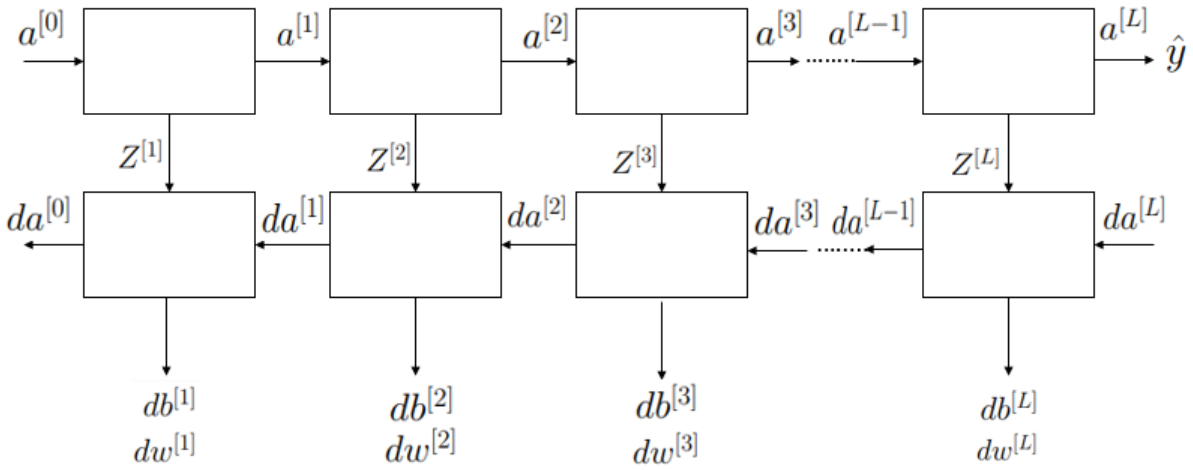


Figure 6: Building Block of Deep Neural Network

$$
\boxed{
\begin{aligned}
W^{[l]} &:= W^{[l]} - dW^{[l]} \\
b^{[l]} &:= b^{[l]} - db^{[l]}
\end{aligned}
}
$$

$$
\begin{aligned}
where, da^{[l]} &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial a^{[l]}} \\
db^{[l]} &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial b^{[l]}} \\
dW^{[l]} &= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial W^{[l]}}
\end{aligned}
$$

Here, a single upper & it's lower block together make a single layer of the network. Each layer has an input i.e. activation of the previous layer. With these inputs the layer completes a forward propagation. The layer gives an output i.e. the gradient of the activation function of it's previous layer completes the previous layer's backward propagation.

4.7 Dimensions of the Parameters

- W : [$n^{[l]}$, $n^{[l-1]}$]
- b : [$n^{[l]}$, 1]
- Z : [$n^{[l]}$, m]
- a : [$n^{[l]}$, m]

where, l = number of layer
$n^{[l]}$ = number of nodes in the l^{th} layer
$Z^{[l]}$ = linear part in the l^{th} layer
$a^{[l]}$ = activation in l^{th} layer

5 Deep Neural Network Training

There are certain practices in Deep Learning that are highly recommended, in order to efficiently train Deep Neural Networks [6].

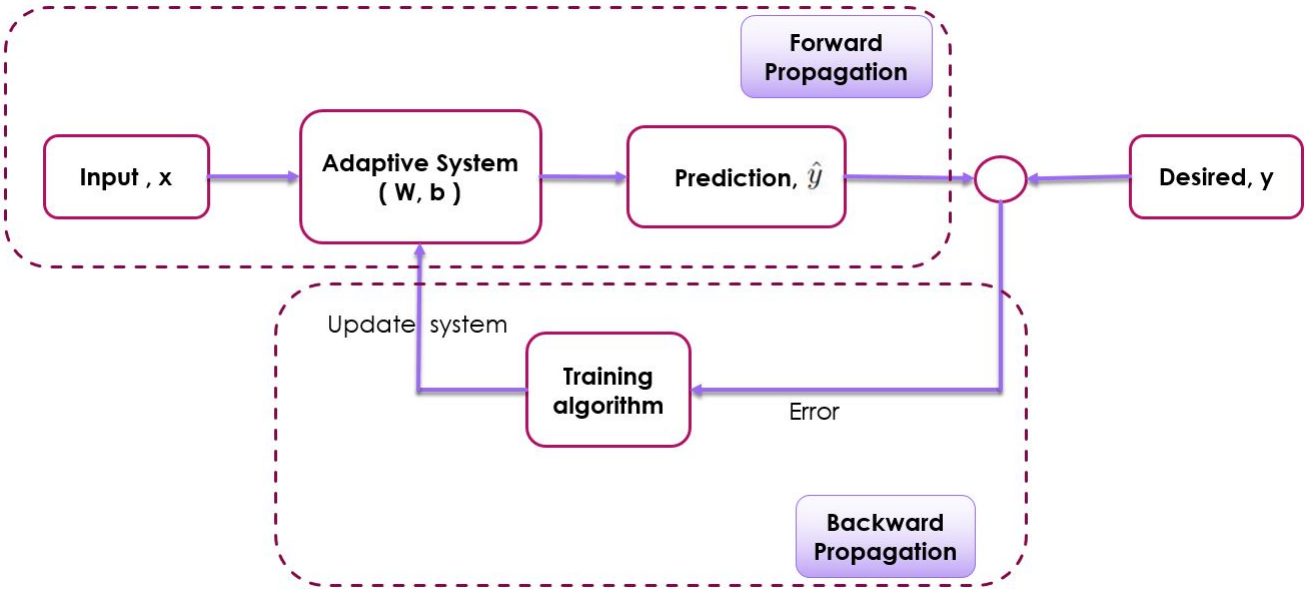


Figure 7: Deep Neural Network Pipeline

5.1 Training data

- Get hands on as large a dataset as possible(DNNs are quite data-hungry: more is better)
- Remove any training sample with corrupted data(short texts, highly distorted images, spurious output labels, features with lots of null values, etc.)
- Data Augmentation - create new examples(in case of images - rescale, add noise, etc.) [6]

5.2 Choose appropriate activation functions

One of the vital components of any Neural Net are activation functions. Activations introduces the much desired non-linearity into the model [6].

5.2.1 Rectified Linear Units(ReLu)

most recent deep learning networks use rectified linear units (ReLUs) for the hidden layers. A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. ReLUs' machinery is more like a real neuron in your body. ReLU activations are the simplest non-linear activation function you can use, obviously. When you get the input is positive, the derivative is just 1, so there isn't the squeezing effect you meet on backpropagated errors from the sigmoid function [7].

$$f(x) = max(x, 0)$$

5.2.2 Softmax

The sigmoid function can be applied easily, the ReLUs will not vanish the effect during your training process. However, when you want to deal with classification problems, they cannot help much. Simply speaking, the sigmoid function can only handle two classes, which is not what we expect. The softmax function squashes the outputs of each unit

to be between 0 and 1, just like a sigmoid function. But it also divides each output such that the total sum of the outputs is equal to 1 (check it on the figure above).Mathematically the softmax function is shown below, where z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z). And again, j indexes the output units, so j = 1, 2, ..., K [7].

$$\sigma(Z)_j = \frac{\exp^{(Z_j)}}{\sum_{k=1}^{K} \exp^{(Z_k)}}$$

5.3 Number of Hidden Units and Layers

Keeping a larger number of hidden units than the optimal number, is generally a safe bet. Since, any regularization method will take care of superfluous units, at least to some extent. On the other hand, while keeping smaller numbers of hidden units(than the optimal number), there are higher chances of underfitting the model [7].

5.4 Weight Initialization

Always initialize the weights with small random numbers to break the symmetry between different units. Furthermore, while using activation functions, if weights are initialized to very large numbers, then the activation function will saturate(tail regions), resulting into dead neurons. If weights are very small, then gradients will also be small. Therefore, it's preferable to choose weights in an intermediate range, such that these are distributed evenly around a mean value [7].

5.5 Learning Rates

This is probably one of the most important hyperparameter, governing the learning process. Set the learning rate too small and your model might take ages to converge, make it too large and within initial few training examples, your loss might shoot up to sky. Generally, a learning rate of 0.01 is a safe bet, but this shouldn't be taken as a stringent rule; since the optimal learning rate should be in accordance to the specific task [7].

6 Learning Method

6.1 Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as **gradient ascent** [8].

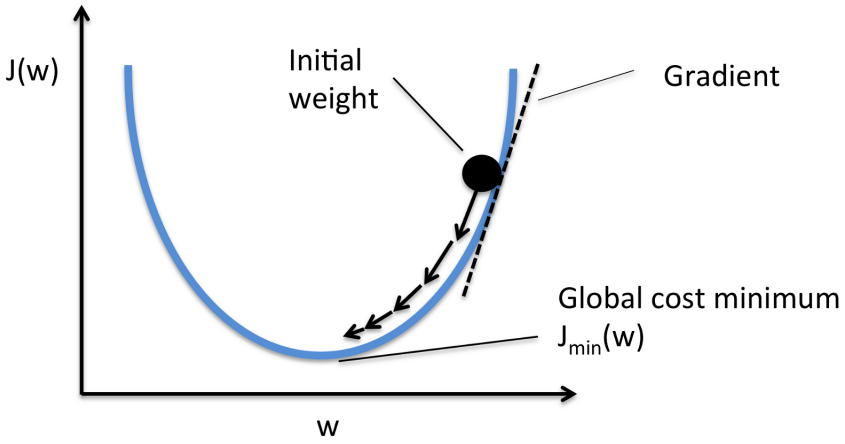


Figure 8: Gradient Descent

6.2 Stochastic Gradient Descent

Stochastic gradient descent (often shortened to SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions. In other words, SGD tries to find minima or maxima by iteration [9].

In stochastic gradient descent, the true gradient of Q(w) is approximated by a gradient at a single example:

$$w := w - \eta \nabla Q_i(w).$$

In pseudocode, stochastic gradient descent can be presented as follows:

1. Choose an initial vector of parameters ω and learning η
2. Repeat until an approximate minimum is obtained:
   (a) Randomly shuffle examples in the training set.
   (b) For $i = 1, 2, ..., n$ do:
      i. $\omega := \omega$ - $\eta \nabla Q_i(\omega)$

7 One Hot Encoding

Many times we will have a y vector with numbers ranging from 0 to C-1, where C is the number of classes. If C is for example 4, then you might have the following y vector which we will need to convert as follows:

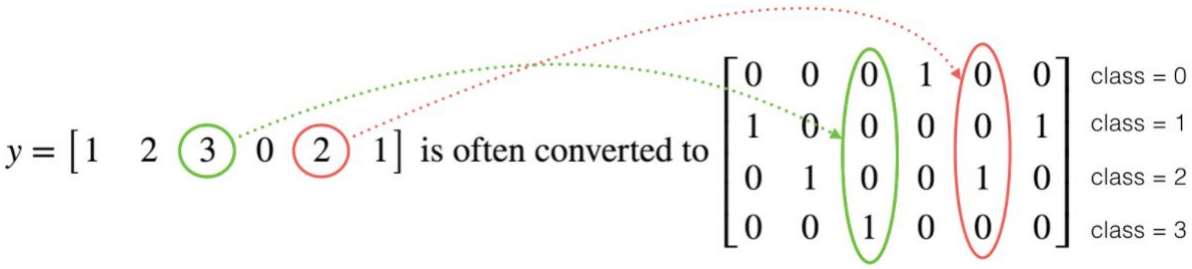


Figure 9: One Hot Encoding

This is called a "one hot" encoding, because in the converted representation exactly one element of each column is "hot" (meaning set to 1).Encode categorical integer features using a one-hot aka one-of-K scheme.

The input to this transformer should be a matrix of integers, denoting the values taken on by categorical (discrete) features. The output will be a sparse matrix where each column corresponds to one possible value of one feature. It is assumed that input features take on values in the range [0, n_values). This encoding is needed for feeding categorical data to many scikit-learn estimators, notably linear models and SVMs with the standard kernels [10].

8 Dataset Analysis

- Bosche Traffic Light Data set contains 5093 images training images and 8334 testing images.
- As we only implemented the classification part of the proposed model [11], we crop the traffic light images from the training and testing set.
- In this paper [11], to detect the false positive of the detector they added background images.
- For this they created the training set by randomly cropping bounding boxes around the ground truth object.
- We did the same and created the training set of 12503 which contains 10698 traffic light images and 1805 background images.
- The test set contains 13486 images.

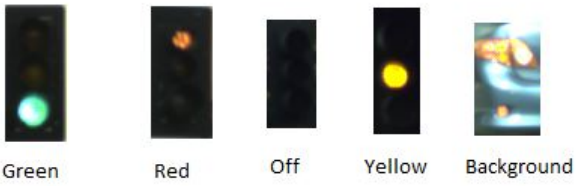


Figure 10: Sample pictures of training data set

9 Our Own Data set

1. Till now we have 50 minutes of video from prospective of the driver of a car.

Figure 11: Our Own Data set

2. From the existing Bosche Traffic Light Dataset we leveled around 3000 brake lights.

Figure 12: Labeled brake light images of Bosche Dataset

10 Classification

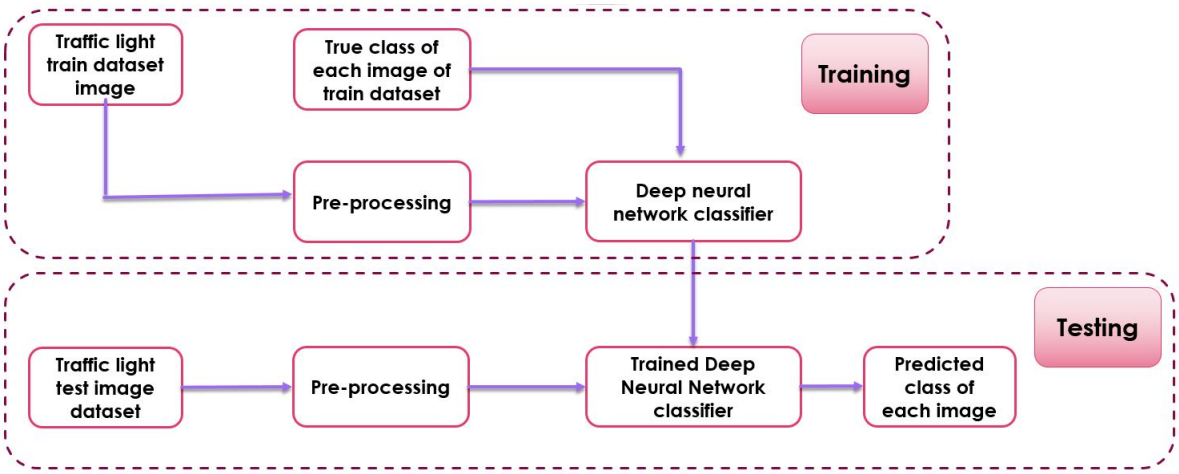


Figure 13: Classification Pipeline

As detection of traffic light is our future plan, for now we implemented classification part. The state of all detected traffic lights needed to be determined. To do so we created a small classification network that differentiates between the different traffic light states and additionally removes false positive. All bounding boxes are re-scaled to 64 × 64 pixels. Overall we created 7496 images for training and 3693 for validation. With the limited samples of labeled data, we restrict our classier training to the "background", "green", "yellow", "red", and "off" classes. The "off" class is necessary for single frame classication since trafc lights often appear as turned off for several frames at a time in camera images.

The model [11] implemented here used convolutional neural network. Here we implemented our network using a deep fully connected neural network. Our model contains 4 weight layers where all these networks are fully connected. All layers use rectified linear units as activation, except for the output layer with a softmax function.

Input	**3*64*64**
Fully Connected:	**units:1024,ReLu**
Fully Connected:	**units:512,ReLu**
Fully Connected:	**units:256,ReLu**
Fully Connected:	**units:128,ReLu**
Fully Connected:	**units:64,ReLu**
Fully Connected:	**units:32,ReLu**
Fully Connected	**units:5,Softmax**

11 Results & Analysis

They [11] trained a classifier to remove false positives and decided between different traffic light states "green", "yellow", "red", "off",and "background". It takes about 0.06 ms to classify 32 samples with their model. With an accuracy of approximately 99% on the validation set they reach 95.1% on slightly translated ground truth data from the test-set. As the largest source of error they [11] identied traffic lights labeled as "off", which are often falsely classied as "background".The difference in accuracy can have multiple reasons,such as slight differences in the types of traffic lights, background, or even lighting.

To remove false positives and differentiates different traffic lights states "green", "yellow", "red", "off",and "background", we trained our classifier. As we only implemented the classification part of our model using deep neural network, we are able to make approximately 93% validation accuracy and 95% test accuracy. Moreover, in our model we need .8 milliseconds to classify a single image whereas they [11] only need .0018 milliseconds which is very slow compared to them. We except a better evaluation and result over test and validation accuracy and prediction time when we will implement convolutional neural network in our model. The evaluation and results are given below: We

	Existing Result	**Our Result**
Validation Result:	**99%**	**93%**
Test Result :	**95.1%**	**95%**
Classification Speed per second:	**.0018ms**	**.8ms**

have got the same accuracy on test set and less accuracy on validation set. Because there is no background images in the test set. When we will complete solution pipeline of our problem, at that time detection part will be implemented. The detector that we will implement won't get 100% accuracy because of detecting false positive images as traffic light. This false positive images will be sent to the classifier. The classifier will try to detect this false positive images as background. As we don't have any flase positive images in our test dataset, we are getting same accuracy as the exitting work. But our validation dataset contains some false positive images. That's why we are getting actual accuracy.

We could find better validation and test accuracy on our classifier but the frequency of the "off" and "yellow" classes in the dataset was really low. There was't even a thousand image of each class of "off" and "yellow".

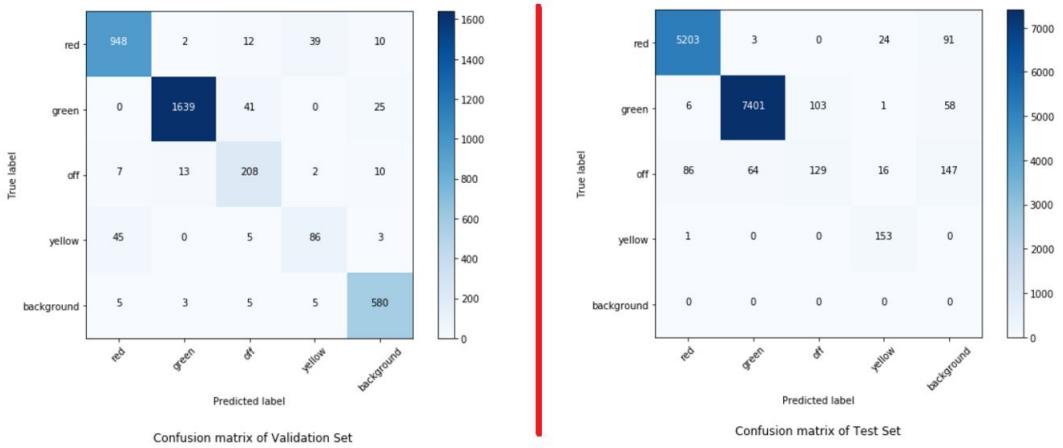


Figure 14: Classification Pipeline

12 Future Works

- We will convert our classification model from Depp Neural Network to Convolutional Neural Network.
- We will implement the detection part of traffic light.
- We will label our own data set for car brake light.
- We will implement detection and classification over our own dataset of car brake light and the existing Bosche Traffic Light Dataset [12].

13 Conclusion

An autonomous car is the ultimate evolutionary goal of developing ADASes - Advanced Driver Assistance Systems, to the point when there's nobody to assist anymore. Computer vision is the most important part for autonomous car. Computer vision techniques are irreplaceable on an autonomous car because it provides a last layer of security against unexpected incidents. We implement the classification of our problem solving pipeline. We will get even better accuracy when we will implement the full problem solving pipeline and implement this classifier with convolution neural network.

References

[1] Wikipedia. Autonomous car—-wikipedia. https://en.wikipedia.org/wiki/Autonomous_car.

[2] The ethics of autonomous cars. https://ethicsofautonomouscars.weebly.com/motivation.html.

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

[4] *CS231n: Convolutional Neural Networks for Visual Recognition.* http://cs231n.github.io/neural-networks-1/.

[5] *Deep Learning Notations.* 2016. https://www.coursera.org/learn/neural-networks-deep-learning/lecture/Z8j0R/binary-classification.

[6] How to train your deep neural network. http://rishy.github.io/ml/2017/01/05/how-to-train-your-dnn/.

[7] Relu and softmax activation functions. https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions.

[8] Wikipedia. Gradient descent — wikipedia, the free encyclopedia, 2017.

[9] Wikipedia. Stochastic gradient descent — wikipedia, the free encyclopedia, 2017.

[10] sklearn.preprocessing.onehotencoder. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.

[11] Libor Novak Karsten Behrendt and Rami Botros. *A deep learning approach to traffic lights: Detection, tracking, and classification.* IEEE International Conference on Robotics and Automation (ICRA) pp. 1370-1377, 2017.

[12] Bosche traffic light dataset. https://hci.iwr.uni-heidelberg.de/node/6132.