



PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA  
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH  
TLEMCEM UNIVERSITY  
FACULTY OF TECHNOLOGY

**Departement of Electrical and  
Electronics Engineering  
Speciality Automation  
Option: Industrial Computer Science**

Thesis for obtaining the Master's degree  
on Automation, option Industrial  
Computer Science

---

**Automatic recognition of road signs:  
contribution to recognition in bad  
weather conditions**

---

**Presented by:**

**Mr Omar BELGHAOUTI**

Defended on 24/09/2020 in front of the jury

<i>PRESIDENT :</i>	Mrs A. Choukchou-Braham	PR, Tlemcen univesity
<i>EXAMINATOR :</i>	Mr M. El Habib Daho	MCB, Tlemcen univesity
<i>SUPERVISOR :</i>	Mrs W. Handouzi	MCB, Tlemcen univesity
<i>CO-SUPERVISOR:</i>	Mr M.C. Benhabib	MCA, Tlemcen univesity

---

2019/2020



# Abstract

In order to improve road safety and significantly reduce the risk of accidents due to poorly visible road signs because of bad weather conditions, numerous works have been undertaken. In this work, we propose a contribution to this issue by taking advantage of the remarkable results of deep ConvNet in computer vision. We propose an automatic recognition system of road signs based on a modified model inspired by LeNet model. The results obtained by comparison of LeNet model and two proposed modified models on the German traffic dataset is about 99 % accuracy which is promising compared to the state-of-the-art results which also showed promising results on classifying traffic signs with bad weather conditions.

**Keywords :** Traffic sign recognition, ConvNet, German Traffic Signs Dataset (GTSD), LeNet.

# *Acknowledgments*

*First of all, I thank the good Lord who brought me help to do this job well.*

*To my grandmother who I wished to see me graduate this year ... I pray for you everyday ... thank you for being there for me by your prayers ... I couldn't be here without you ... I hope we can all meet in the good heaven.*

*To my parents who were always by my side ... Thank you for believing in me and also I always aspire to make you proud as long as I do my work.*

*This work would not be as rich and would not have been possible without the help and supervision of **Mrs W. HANDOUZI** and **Mr C. BENHABIB** ... I thank them for the quality of their exceptional supervision, for their patience, their thoroughness and their availability during my preparation of this thesis.*

*I would also like to express the honor to the members of the jury, **Mrs A. CHOUKCHOU-BRAHAM** and **Mr M. EL HABIB DAHO** and for having accepted to examine and evaluate my work.*

*At the end, I thank all people who believed in me to realize this work.*

# Contents

<b>Abstract</b>	<b>II</b>
<b>Acknowledgments</b>	<b>III</b>
<b>List of Abbreviations</b>	<b>IX</b>
<b>General introduction</b>	<b>1</b>
<b>I State of the art</b>	<b>2</b>
I.1 Introduction . . . . .	3
I.2 Conclusion . . . . .	6
<b>II Theoretical notions</b>	<b>7</b>
II.1 Introduction . . . . .	8
II.2 Machine learning . . . . .	8
II.2.1 Machine learning methods . . . . .	8
II.2.2 Regression . . . . .	9
II.2.3 Classification . . . . .	10
II.3 Artificial neural network . . . . .	10
II.3.1 Processing unit (Perceptron) . . . . .	11
II.3.2 Interconnections (Weights) . . . . .	11
II.3.3 Operations . . . . .	12
II.3.4 Update . . . . .	12
II.4 How neural networks works . . . . .	12
II.4.1 Feedforward . . . . .	14
II.4.2 Backpropagation . . . . .	16
II.5 Deep learning . . . . .	17
II.6 Architectures of deep learning . . . . .	17
II.6.1 Convolutional neural networks . . . . .	18
II.6.2 Main components in a ConvNet . . . . .	19
II.7 Conclusion . . . . .	24
<b>III Methodology</b>	<b>25</b>
III.1 Introduction . . . . .	26
III.2 German Traffic Signs Dataset (GTSD) . . . . .	26
III.3 Data augmentation . . . . .	29
III.4 Preprocessing stages . . . . .	29

---

III.5 The proposed deep convnet architectures . . . . .	31
III.6 Conclusion . . . . .	35
<b>IV Results &amp; discussions</b>	<b>36</b>
IV.1 Introduction . . . . .	37
IV.2 Preliminary results . . . . .	37
IV.2.1 LeNet model . . . . .	37
IV.2.2 Modified model 1 . . . . .	38
IV.2.3 Modified model 2 . . . . .	38
IV.2.4 Discussion . . . . .	39
IV.3 Testing on new images . . . . .	39
IV.3.1 Discussion . . . . .	40
IV.4 Testing on images with bad weather conditions . . . . .	42
IV.4.1 Discussion . . . . .	45
IV.5 Conclusion . . . . .	45
<b>General conclusion</b>	<b>46</b>
<b>Bibliography</b>	<b>47</b>

# List of Figures

I.1	Traffic sign detection and recognition system . . . . .	3
I.2	German traffic sign dataset . . . . .	4
I.3	Color segmentation . . . . .	4
I.4	Multi scale deconvolution network . . . . .	5
II.1	Machine learning methods . . . . .	8
II.2	Linear regression model . . . . .	9
II.3	Artificial neural network . . . . .	11
II.4	Plain vanilla "multilayer perceptron" . . . . .	13
II.5	Input layer to first hidden layer . . . . .	14
II.6	Sigmoid function . . . . .	15
II.7	ReLU function . . . . .	15
II.8	Convolutional neural network . . . . .	19
II.9	Input image of size $4 \times 4 \times 3$ . . . . .	20
II.10	Convoluting a $5 \times 5 \times 1$ image with a $3 \times 3 \times 1$ kernel to get $3 \times 3 \times 1$ convolved features . . . . .	20
II.11	Kernel movement . . . . .	21
II.12	Convolution operation on a $M \times N \times 3$ image matrix with a $3 \times 3 \times 3$ kernel . . . . .	22
II.13	Types of pooling . . . . .	23
II.14	Flattening . . . . .	24
III.1	Distribution of the train set . . . . .	26
III.2	Sample of augmented data . . . . .	29
III.3	Input image . . . . .	30
III.4	Conversion to grayscale . . . . .	30
III.5	Histogram equalization . . . . .	31
III.6	LeNet base model . . . . .	32
III.7	Modified model 1 architecture . . . . .	33
III.8	Modified model 2 architecture . . . . .	34
IV.1	Results for LeNet model . . . . .	37
IV.2	Results for modified model 1 . . . . .	38
IV.3	Results for modified model 2 . . . . .	38
IV.4	Confusion matrix . . . . .	40
IV.5	Prediction on a stop panel image . . . . .	41
IV.6	Prediction on another stop panel image . . . . .	41
IV.7	Prediction on slippery road sign with bad weather conditions . . . . .	43

---

IV.8 Prediction on stop sign with bad weather conditions . . . . . 44



# List of Tables

III.1	GTSDDB classes with labels . . . . .	27
III.2	Number of parameters in each used architecture . . . . .	35
IV.1	The distribution of GTS dataset . . . . .	37
IV.2	Preliminary results for all models . . . . .	39
IV.3	Predictions on the slippery road sign with bad weather conditions . . . . .	42
IV.4	Prediction on stop sign with bad weather conditions . . . . .	43

# List of Abbreviations

**SVM** : Support Vector Machine.

**TSDR** : Traffic Sign Detection and Recognition.

**RGB** : Red, Green and Blue.

**LSS** : Local Self-Similarity.

**HOG** : Histogram of Oriented Gradient.

**HSI** : Hue, Saturation and Intensity.

**AUC** : Area Under the ROC Curve.

**GTSD** : German Traffic Signs Dataset.

**SURF** : Speed Up Robust Features.

**ANN** : Artificial Neural Network.

**HSV** : Hue, Saturation and Value.

**KNN** : K-Nearest Neighbours.

**MDN** : Multi-Scale Deconvolution Network.

**YOLO** : You Only Look Once.

**ML** : Machine Learning.

**AI** : Artificial Intelligence.

**MSE** : Mean Square Error.

**MNIST** : Modified National Institute of Standards and Technology.

**ReLU** : Rectified Linear Unit.

**SGD** : Stochastic Gradient Descent.

**CNN/ConvNet** : Convolutional Neural Network.

**DL** : Deep Learning.

**DCNN** : Deep Convolutional Neural Network.

**OCR** : Optical Character Recognition.

**GAN** : Generative Adversarial Networks.

**NN** : Neural Network.

**FC** : Fully Connected.

# General introduction

Road accidents cause the deaths of over 1.35 million people per year, according to the World Health Organization [1]. The causes of this frightening number are numerous. Some are listed in the contribution of P Mikoski et al [2], like distracted driving, poor visibility at night, drivers' age, etc. [2].

Given the frightening number of road deaths, the researchers took advantage of the rise of artificial intelligence and deep learning in computer vision to develop driver assistance applications and help reduce the number of accidents due to poor vision [3], [4]. So, to improve road safety and significantly reduce the risk of accidents due to poorly visible road signs, numerous works have been undertaken in this task.

Computer vision consists on the detection, localization and recognition of different target objects for different applications. One of the most used architectures in it is the convolutional neural networks (ConvNet). They consist of several layers of extraction and learning of relevant parameters based on the concept of convolution. A well-known architecture is LeNet [5] it is used in our work.

The goal of our project is to make a ConvNet model to recognise road signs even in bad weather conditions.

This memoir is divided into four chapters. In the first chapter we will present the related work on traffic sign detection and recognition. The second chapter will be devoted to theoretical notions where we will see in details the applications of machine learning and deep learning. Also, we present the functionality of an artificial neural network and of a convolutional neural network.

In the third chapter we will introduce our methodology to face the problem of recognising road signs for the German traffic signs dataset. We will see the data augmentation and image preprocessing techniques in detail and then we will present all three ConvNet architectures that are going to be used in the training process. The final chapter will be dedicated the training and testing result where we will discuss the preliminary results and from there we will take the best model out of those three. After that, using the best model we will test it on new images and also on images with artificial bad weather conditions to discuss the results of normal images and images with some disturbances.

This work ends with a general conclusion where we will identify some perspectives by the obtained results.

# Chapter I

## State of the art

## I.1 Introduction

In literature, many studies exist with the purpose of traffic sign detection and recognition. In this section, we focused our attention only in papers published the last five years focused on this topic.

An Automatic Traffic Sign Detection and Recognition System Based on Color Segmentation, Shape Matching, and SVM was studied in the work of Wali [6]. He proposed to build an efficient TSDR system based on a dataset of Malaysian signs. The images were captured in different weather conditions by an on-board camera and image preprocessing was done with the use of the RGB color segmentation. The recognition process is carried out by SVM with a packed kernel that is first used for the classification of traffic signs. An accuracy and processing time about 95 % 0,43s respectively were obtained.



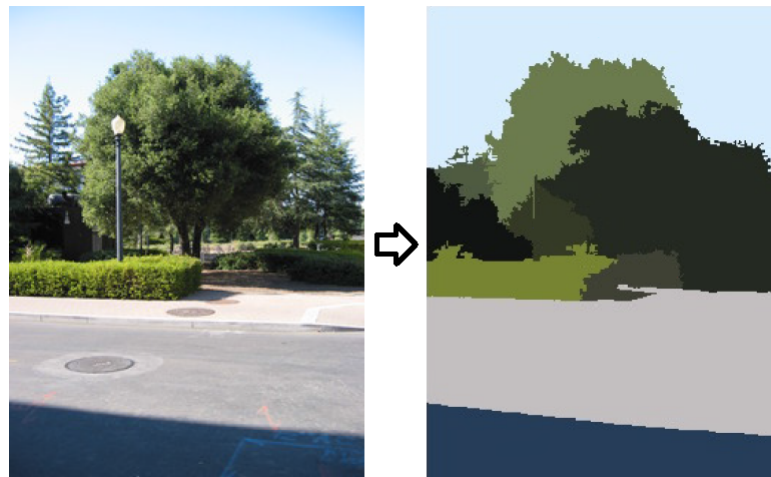
Figure I.1: Traffic sign detection and recognition system [7]

In 2016, Ellahyani [8] proposed a new method of detecting and recognizing traffic sign (TSDR) in three main steps. The first step divides the image on an HSI color space threshold. The second step detects traffic signs through the first step in the processing of blobs extracted. The last one recognizes the traffic signs detected. Main contributions of this work are as follows, first, in the second step they proposed to use invariant geometric times to classify the forms rather than the algorithms of machine learning. Secondly, new features were proposed for recognition inspired by the existing features. In combine with Local Self-Similarity (LSS) features, the Histogram of Oriented Gradient (HOG) features is extended to the HSI color area to obtain the descriptor they used in their algorithm. As a classifier, the new descriptor tested random forest and support machine (SVM) classifiers. Results achieves 94.21 % AUC on the GTSD (See figure I.2) at a processing rate 8–10 frames/s.



Figure I.2: German traffic sign dataset

Fuzzy segmentation approach and artificial neural network classifiers were used to Traffic sign detection and recognition respectively [9]. In 2017, this represents a new approach to the TSDR system where traffic signal detection is performed with a fuzzy rules of color segmentation (Figure I.3) and recognition is achieved using Speed Up Robust Features (SURF) which is trained by Artificial Neural Network (ANN) classifier. The detection step, area of interest, is divided by a set of fuzzy rules that will be processed after the filtering process of unwanted area depending on the hue and saturation values in the HSV pixel color space. A SURF classifier was trained using ANN. The system proposed simulated images captured in various lighting conditions on offline road scene. The accuracies obtained on detection and recognition were 95 %, 97 % respectively.

Figure I.3: Color segmentation  
[10]

In the same year, Islam and Raj [11] proposed a real-Time (Vision-Based) road sign recognition system using an Artificial Neural Network (ANN). It is based on two steps, the detection is performed one and recognition is performed the other. In the first step an algorithm was developed and tested for the hybrid color segmentation. The second step is the introduction of a robust, individually customized extractive method for a road sign recognition approach for the first time. Finally, a multilayer artificial neural network has been established to identify various road signs and to interpret them. Results were promising 99 % recognition accuracy and 0.12s processing time per image.

In the aim of developing an automatic measurement of the traffic sign with digital segmentation and recognition, Khalid et al [12] proposed initially, the threshold value estimation using the image's correlation property. A segmentation algorithm using estimated threshold and morphologic operations followed by an enhancement procedure is developed to obtain red and blue traffic signs which provides a greater number of potential signs to the net result. In addition, the remaining regions are filtered using non-potential regions in terms of statistical measures. Also, detection is carried out by using the SVM-KNN classification on the basis of the HOG features. The denoising approach with the weighted fusion of KNN and SVM is employed to reduce the false positive by improving the performance of the proposed algorithm. Results were promising compared to the state-of-the-art methods.

In 2018, Pei et al [13] proposed a multi-scale MDN (Multi-Scale Deconvolution Network) which flexibly combine multi-scale convolution neural network as the following figure shows. Also, they used the deconvolution subnetwork to tackle problems by taking a lot of time to compute complicated algorithms and low detection images. The proposed system aims to localized and detection Traffic Sign. It has obtained about 85 % accuracy.

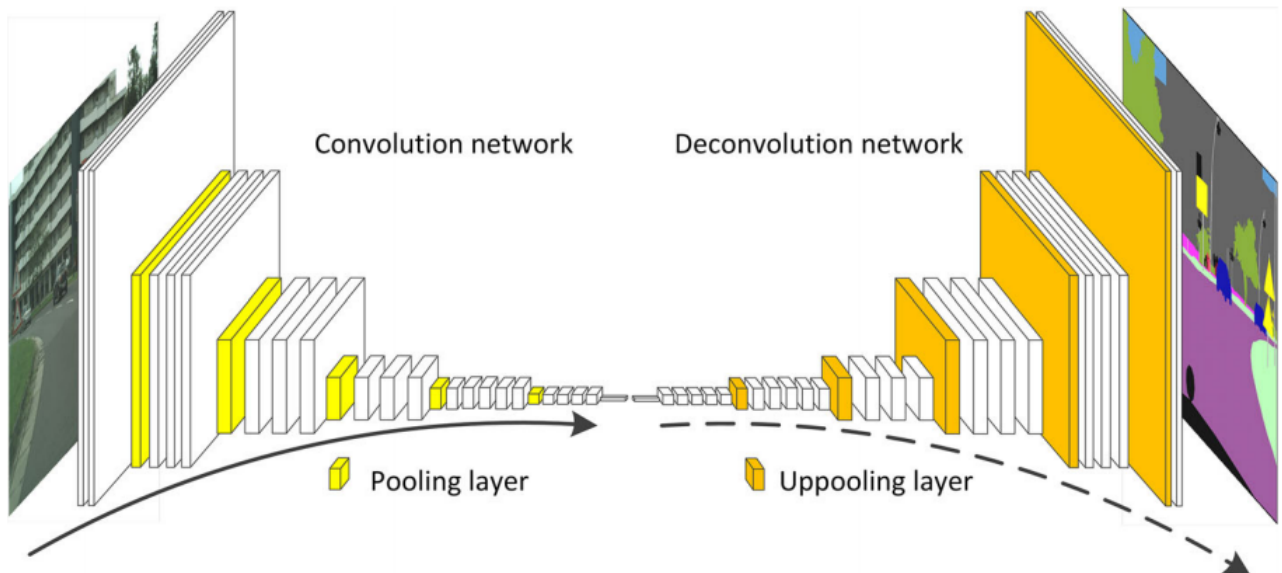


Figure I.4: Multi scale deconvolution network [14]

In 2019, several studies were published aiming traffic sign detection and/or recognition. We will cite two of them.

The first proposed by Jose et al [15] in which a new approach is presented that combines Viola-Jones frame working with deep learning. The system is designed and tested under Indian road conditions for its effectiveness. It was observed that even in the presence of unstandardized signals, the designed system is able to detect correct traffic signals.

The second work, a new method of detecting sign fully-data-driven by working in a customized color space with the non-linear separation of training data in mind was proposed. Also, an hy-



brid, multi-scale radial extraction method to recognize the content of traffic signs was proposed. Features are drawn by intelligently working at multiple scales and by giving the most informative part of the pictogram importance. An effective feature selection strategy based on feature interaction to get rid of irrelevant and redundant attributes in a long hybrid feature vector was used. In addition to a benchmark dataset, an automatically collected data set from the largest national highway in Pakistan (N5 road, for example) is also used to test the proposed traffic sign detection and recognition algorithms which represent deterioration typical of developing world [16]. The two works obtained good accuracies 92 % on Indian dataset and 90 % on GTSD respectively.

## I.2 Conclusion

The reported approaches achieved good accuracy rates for traffic sign detection and recognition, which are validated using different methods and on different datasets. However, these recognition rates are strongly dependent on the application context without taking under consideration for example bad weather conditions. Also, Deep ConvNet and data augmentation for datasets can improve results of classification in bad conditions. For this, the next chapter will be devoted on theoretical notions on deep learning and neural networks.

## Chapter II

### Theoretical notions

## II.1 Introduction

Artificial intelligence (AI) has been the center of interests in a decade from now, many problems has been solved using machine learning (ML) and deep learning (DL) techniques. Data has been grown also and that what makes those techniques relevant to deal with nowadays problems.

## II.2 Machine learning

Machine learning (ML) is an artificial intelligence (AI) application that allows systems to learn from experience automatically and to improve without explicitly programming. Machine learning is focused on developing computer programs that can access and use data for their own purposes [17].

In order to look for data patterns and make better choices in the future, we start the learning process by observations or information, such as examples, direct experience or instruction, based on the examples we provide. The main objective is to allow computers to automatically learn and adjust actions without human intervention or support.

### II.2.1 Machine learning methods

Algorithms for machine learning are often classified as supervised, unsupervised, semi-supervised or reinforcement learning. The following figure show the four main classes in ML.

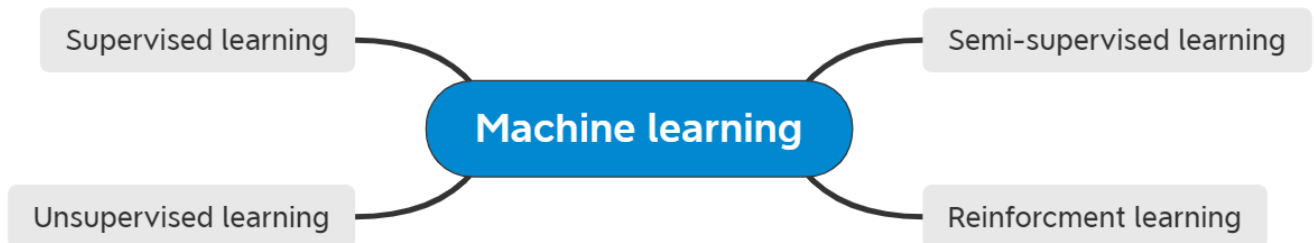


Figure II.1: Machine learning methods

1. **Supervised machine learning algorithms** use labeled examples to predict future events to apply what was previously learned to new data. The learn algorithm produces a deduced function to predict the output values, starting with an analysis of a knowledged data set. After sufficient training, the system can provide targets for any new input. The learning algorithm can also compare its output to the right output and detect errors so that the model is modified accordingly.
2. **Unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can deduce

a function from unlabeled data that describes a hidden structure. The system does not provide the correct output, but it examines the data and can draw inferences from datasets to describe unlabeled structures.

3. **Semi-supervised machine learning algorithms** fall into somewhere between supervised and unsupervised learning, as both labeled and unlabeled training data are used, typically for the use of a small number of marked data and a great deal of unlabeled data. The systems using this method can significantly improve the accuracy of learning. Usually, semi-supervised learning is chosen when it needs skilled and relevant resources to learn from the acquired data. Otherwise, it usually does not require additional resources to acquire unlabeled data.
4. **Reinforcement machine learning algorithms** is a learning process that interacts and discovers errors or rewards with its environment. The most relevant characteristics of enhancement learning are trial and error search and delayed rewards. In order to maximize its performance, machines and software agents can automatically identify the optimal behavior in a given context. For an agent to learn what action is best, a simple feedback on the reward is required.

Machine learning has a lot of tasks, but we are interested in supervised tasks which means we have to bring a labeled dataset to train our model. There for we need to have an overview on supervised learning tasks.

## II.2.2 Regression

A task of supervised learning that helps us to predict on linear datasets. This data can consists of random points that on average may follow a pattern (see figure II.2). There are many types of regression like logistic regression and linear regression.

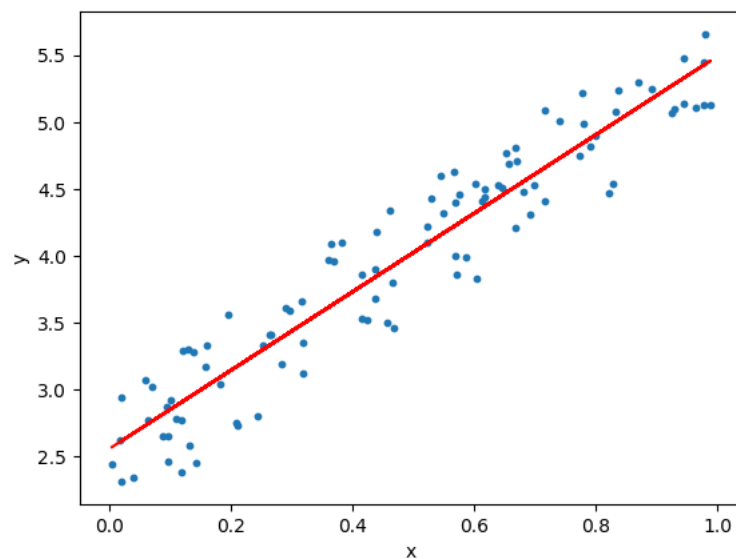


Figure II.2: Linear regression model

The linear model (which is represented in a red line in the last figure) is the result of linear regression, it allows to make predictions on new data.

To obtain this model we need to minimize the cost function that also represents the error between the actual and the predicted output. This error is generally written as the Mean Square Error (MSE) which is defined as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \tilde{Y}_i)^2 \quad (\text{II.1})$$

Where:

$n$ : number of all actual outputs.

$Y_i$ : actual outputs.

$\tilde{Y}_i$ : predicted outputs.

This error will decrease in every iteration of the optimization process adjusting the parameters of the model.

### II.2.3 Classification

Another supervised learning task is classification predictive modeling which approximates a mapping function that predicts a particular observation class or category. In other words, unlike linear regression problem which find the best fitted model for the experimental points, in classification we actually classify those points which is more interesting in real world problems.

After knowing the general points in machine learning, we can go now to see how it actually work by knowing the basic things in artificial neural networks.

## II.3 Artificial neural network

An artificial neural network (see figure II.3) is inspired from biological neural networks that can be found in human brains. It can be defined with the following points [18]:

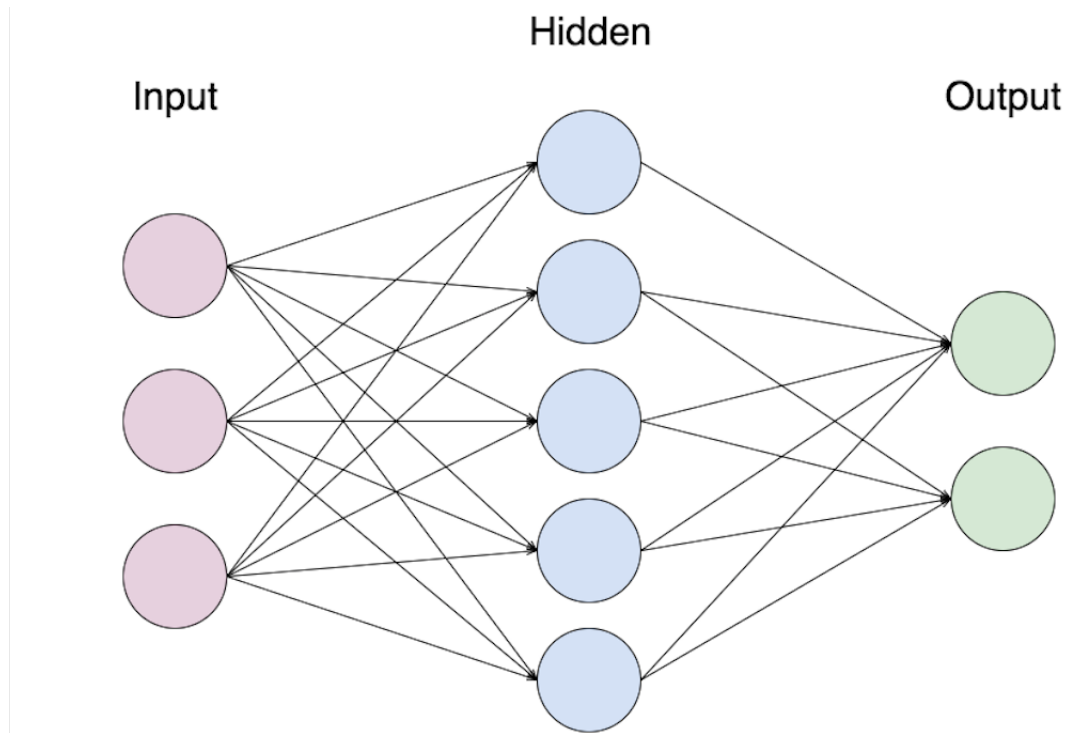


Figure II.3: Artificial neural network

### II.3.1 Processing unit (Perceptron)

A highly simplified model for the structure of the biological neural network can also be viewed as an artificial neural network (ANN). An ANN consists of the processing units interconnected. A processing unit's general model consists of the summing part and an output part. The summing part receives  $N$  values, measures and calculates a weighted sum of each value. The weighted value is called the activation value. The output part generates an activation value signal. The weight sign for each input determines whether the input is excitatory (positive weight) or inhibitory (negative weight). Input and output may be deterministic, stochastic or fuzzy as well.

### II.3.2 Interconnections (Weights)

In an artificial neural network, several processing units are interconnected according to some topology to perform a task of pattern recognition. Consequently, data from other processing unit outputs and/or from external sources may come from a processing unit. Several units, including itself, may be given each unit output. The output of one unit received by another unit depends on the strength of the connection between the units and is shown in the weight of the link. If  $N$  units are found in a given ANN, every unit has a single value and a single output value at any instant of time. The set of the  $N$  network activation values defines the network activating status at that instant. The network status may be described in a  $N$ -dimensional space by a discrete or continuous point, depending on the discrete or continuous nature of the activation and output values.

### II.3.3 Operations

Every ANN unit receives inputs from other connected units and/or external sources in operation. At a given moment, a weighted sum of the inputs is calculated. The value of the activation determines the actual output of the output unit, i.e. the output of the unit. The output values and other external inputs in turn determine the activation and output of other units. The dynamics of activation determine the activation value of all the units, i.e. the network activation state as a time function. The dynamics of activation also determine the dynamics of the network output. The set of all activation states defines the network's activation state space. Dynamics of activation determine the path of the states within the network state space. The activation states determine the short-term memory function of the network for a given network, defined by the units and their interconnections with appropriate weight.

In general, the activation dynamics are followed with an external input to retrieve a pattern that is stored in a network. In order to store a pattern in a network, the weights of the connections in the network must be adjusted. A weight vector is the set of every weight on all connections in the network. The weight space is defined by a set of all possible vectors. The synaptic weights of the network determine the weight vector depending on the time when these weights change. To adapt the weights to store the given patterns on the network, synaptic dynamics are followed. The weight adjustment process is called learning. The final set of weight values, once the learning process is completed, corresponds to the long-term network memory function. The process of updating each of the weights is known as a learning law or learning algorithm.

### II.3.4 Update

In implementation, there are several options available for both activation and synaptic dynamics. In particular, all units can be updated synchronously in their output states. In this case, the activation values for all units are determined simultaneously, assuming a certain output level. The new network performance status is extracted from the activation values. On the other hand, each unit is sequentially modified in an asynchronous update, each time taking into account the current network performance status. The performance status for each unit can be calculated either deterministically or stochastically from its activation value.

The activation dynamics and changes of the biological neural network are far more complex in action than the models mentioned above. According to the pattern recognition task which is needed, the model ANN and the activation and synaptic dynamic equations are constructed.

## II.4 How neural networks works

Now after knowing the basic points in an ANN, we can talk now about how the neural network works. Supposing we have a plain vanilla ANN for MNIST dataset<sup>1</sup> as shown in the next figure:

---

<sup>1</sup>The MNIST dataset (Modified National Institute of Standards and Technology database) is a large dataset of handwritten digits that is commonly used for training various image processing systems.

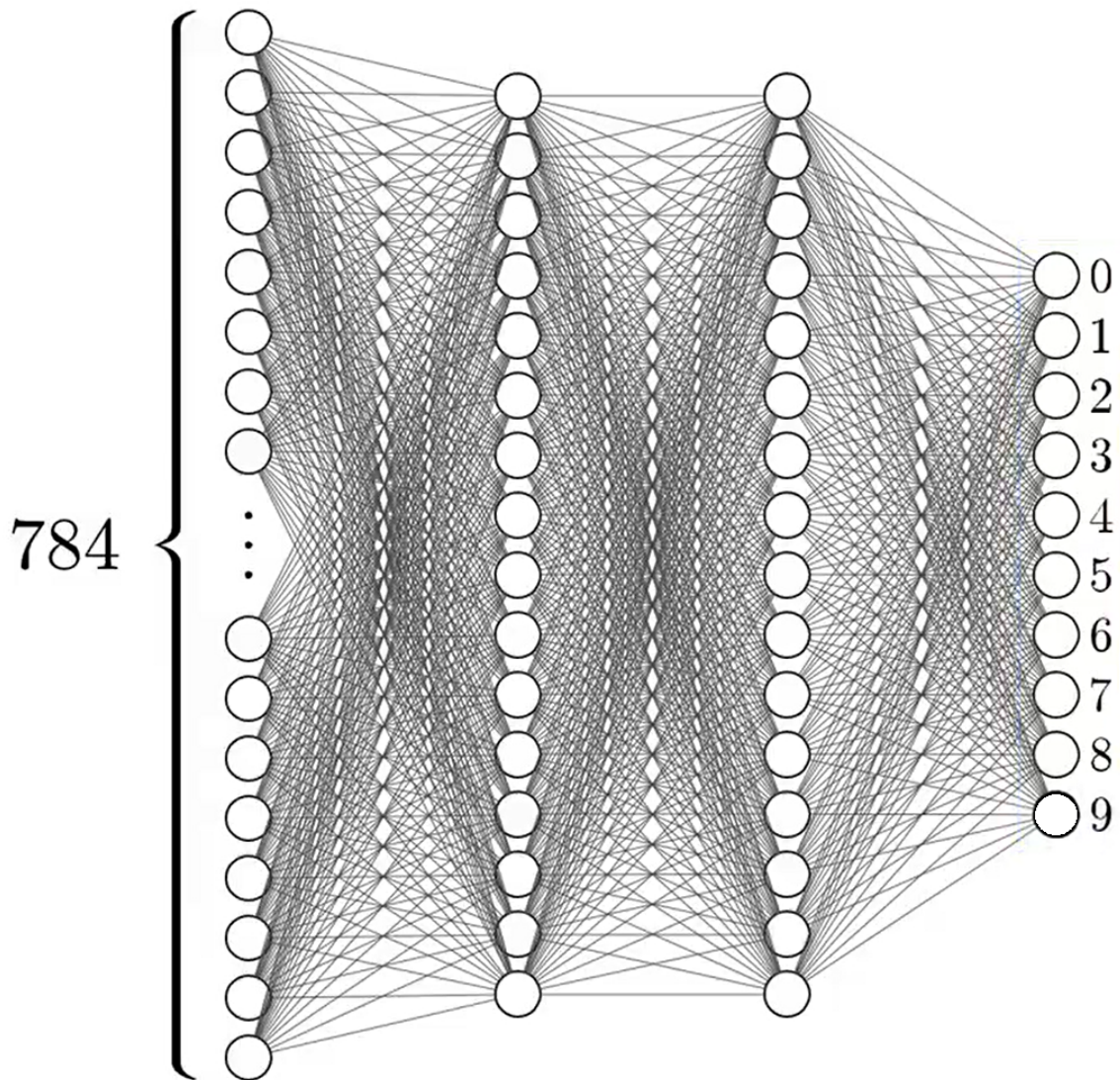


Figure II.4: Plain vanilla "multilayer perceptron"

So a plain vanilla neural network is a basic network full with neurons (perceptrons), and since we are using MNIST dataset which is bunch of handwritten images of  $28 \times 28$  pixels, the network will start with an input layer of 784 neurons ( $28 \times 28 = 784$ ). Each one of these networks holds a number that represents the gray scale value of the corresponding pixel (ranging from 0 "black" up to 1 "white").

The values that each neuron holds it's called its **activation**. And as we can see in the figure II.4, the last layer which is called the output layer has 10 neurons, each one represents one of the digits, the activation in these neurons represents the prediction value of how much this network thinks for a given image corresponds with a given digit. There's also a couple layers between the input and output layer called the **hidden layers**.



## Note

In any kind of neural network, the activation of one layer determines the activation of the next layer.

We know what's the purpose of the input and the output layer, however, the question arises on the hidden layers "Why there are hidden layers?". In our example on predicting a digit, one image of a digit contains **features** which are the components that represent that digit (for example curves, lines and circles), in the second hidden layer before the output layer, we hope that each neuron will holds one those components. As for the first hidden layer, each neuron will holds the sub-components of those curves and lines and circles (for example a circle is a group of 4 curvy lines).

### II.4.1 Feedforward

Now after knowing how the neural network is structured, we can start assigning the parameters which are the activation and the weights in each connection, we compute their weighted sum. The next figure helps understanding what is happening in one neuron of the first hidden layer:

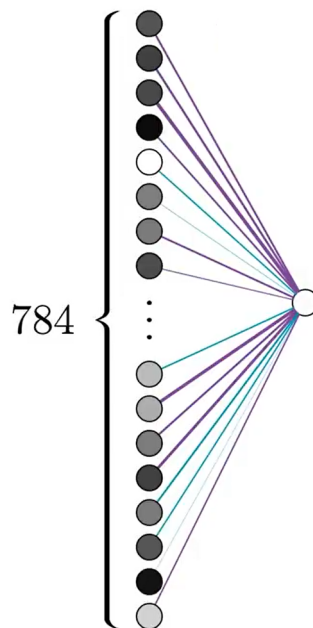


Figure II.5: Input layer to first hidden layer

Each neuron in the input layer holds its activation "a" and each connection from the input to that neuron represent the weight "w", so the weighted sum is written as follows:

$$w_1.a_1 + w_2.a_2 + w_3.a_3 + \dots + w_n.a_n \quad (\text{II.2})$$

Now when we compute this weighted sum, it can be any number, but as we spoke above, each neuron should hold a number between 0 and 1, and here comes the activation function that convert that weighted sum to a percentage value (0 to 1). A common function that does this is

called **Sigmoid** (See figure II.6), which is can be represented with the following equation:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (\text{II.3})$$

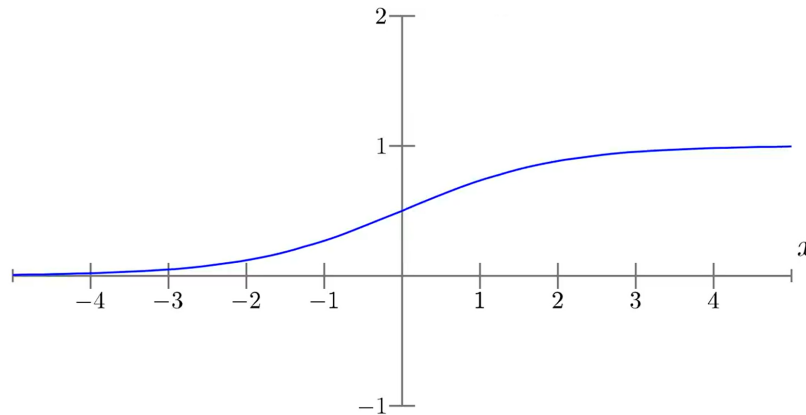


Figure II.6: Sigmoid function

### Note

There are many activation functions, the other most used one is **ReLU** (Rectified Linear Unit) (See figure II.7), it makes the learning process faster than using Sigmoid. It can be represented with the following equation:

$$\text{ReLU}(a) = \max(0, a) \quad (\text{II.4})$$

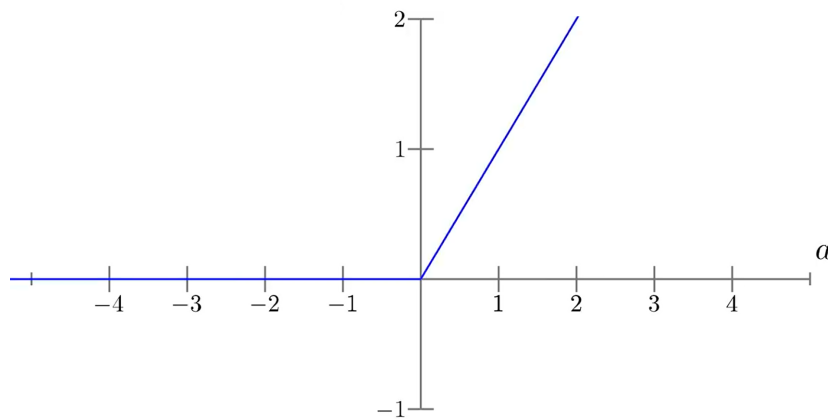


Figure II.7: ReLU function

So the activation here is basically a measure of how positive the weighted sum is. More interestingly is that we can make the neuron active only when the sum is bigger than a specific value which is called the **bias**, so before we apply the activation function we can add a bias "b" for it as follows:

$$\sigma(w_1 \cdot a_1 + w_2 \cdot a_2 + w_3 \cdot a_3 + \dots + w_n \cdot a_n + b) \quad (\text{II.5})$$

And that is just one neuron in the first hidden layer, all other neurons will have the same procedure, and each one of them will have its specific weighed sum and bias. And the same applies for the rest layers.

We can rewrite the equation II.5 in a matrix format (for the first neuron in the hidden layer):

$$a_0^{(1)} = \sigma(w_{0,0}.a_0^{(0)} + w_{0,1}.a_1^{(0)} + \dots + w_{0,n}.a_n^{(0)} + b_0) \quad (\text{II.6})$$

### Note

This writing  $a_0^{(1)}$  means the activation for the first neuron in the first hidden layer, and  $w_{0,n}$  means the weights from the input layer to that specific neuron.

So we can write the following for all the neurons in the first hidden layer:

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = W.a^{(0)} + b \quad (\text{II.7})$$

And now those weighted sums need to be activated in order to get all the activation for the first hidden layer:

$$a^{(1)} = \sigma(W.a^{(0)} + b) \quad (\text{II.8})$$

## II.4.2 Backpropagation

At first, the wights and biases are initialized randomly, as it will make wrong prediction and then it will try to correct that by minimising a **cost function**, as we know of that the MNIST dataset has labeled handwritten digits, so the cost function will be the differences between the wrong predictions and the corrected predictions from the dataset.

This cost function have the weights and biases as parameters (See equation II.9), because we want to have the best parameters by minimizing this function in order to have the corrected predictions, this functions as thousands of parameters, it is so complex that we can't even visualize it.

$$C(w, b) = \Sigma(\text{predictions} - \text{corrected\_labels})^2 \quad (\text{II.9})$$

Since this function is complex, it can have many minimum values, the best parameters corresponds to the **minima** of this function. To find this minimum values, we have to introduce the **gradient descent**. So the algorithm to find the minmum is so simple that it can be written in 3 steps:

1. Compute the gradient of the cost function ( $\nabla C(w, c)$ ).
2. Taking a small step in  $-\nabla C(w, c)$  direction.
3. Repeat it until we find the minimum.

That small step is called the **learning rate** which can be chosen as we want, but commonly it's a very small value so we can't surpass the minima, and also it affects the training time which makes it longer to find the minima.

This algorithm is called the **backpropagation** which is the core algorithm on how neural networks learns. The important thing here is that during this process, a step "epoch" corresponds to a walk-through all the dataset which means in order to adjust the weights and biases for each epoch we should average the cost function to all the dataset then we can apply the backpropagation algorithm to adjust these parameters.

Surely, these parameters will not perfectly be adjusted in one epoch, so we should add more epochs hopefully to obtain the best parameters that corresponds to the global minima of the cost function.

### Note

In practice, this algorithm takes a long time to add up the influence of every single training example in every epoch. What is done instead is that the dataset is randomly shuffled and then divided to bunch of **mini batches**, each one of them contains the same amount of training examples, then a step is computed according to one mini batch (backpropagation applied for one mini batch), this gives a significant computational speed up on the training process. This technique is called "Stochastic gradient descent (SGD)".

## II.5 Deep learning

Deep learning has developed as an effective machine learning method that takes in numerous layers of features or representation of the data and provides state-of-the-art results. The application of deep learning has shown impressive performance in various application areas, particularly in image classification, segmentation and object detection. Recent advances of deep learning techniques bring encouraging performance to fine-grained image classification which aims to distinguish subordinate-level categories. This task is extremely challenging due to high intra-class and low inter-class variance [20].

Deep learning permits computational models consisting of multiple processing layers to learn data representations with various abstraction levels. These methods have enhanced the state of the art in language recognition, visual objects recognition, object detection and many other fields such as medicines and genomics dramatically [3]. Deep learning discovers a complex structure in large sets of data by use of the backpropagation algorithm to show how a computer will adjust its internal parameters used to determine the representation of the previous layer on each layer.

## II.6 Architectures of deep learning

DL and deep convolution neural network (DCNN) have dramatically upgraded the performance beyond the state of the art in the above fields, compared to the conventional machine learning (ML) techniques, such as support vector machine (SVM) [22] and Naive Bayes [23]. It

takes advantage of extracting higher-level features directly from the raw data. There are several advancements of DL that make this model more reliable and adaptive. For instance, in computer vision, a new optical character recognition (OCR) engine [24] is introduced in maps, through which we can identify the street as well as the store signs. Another one is generative adversarial networks (GAN) [25] which enable to tackle the problem of unsupervised learning. Furthermore, there is a task known as visual reasoning, where neural network (NN) is used to answer a question, with the help of a photograph, and so on.

One of the most used architecture is Convolutional Neural Network (CNN), it is widely used especially in image classification problems.

### II.6.1 Convolutional neural networks

The convolutional neural Network (ConvNet/CNN) is a Deep Learning algorithm that allows an image to be captured, attaches importance to various aspects / objects of the image (learnable weights and biases) and is capable of distinguishing between them. ConvNet requires much less pre-processing than other classification algorithms. Although hand-made filters are generated in primitive methods and are trained enough, ConvNets can learn these filters [33].

The ConvNet's functionality is similar to the connectivity model of neurons in the human brain and was influenced by the visual cortex organization. Individual neurons only respond to stimuli in a specific visual field area known as the sensing field. The whole area of vision is filled by a series of these fields.

It was first introduced by Fukushima [26] in 1980, and it had wide variety of activity recognition [27], sentence classification [28], text recognition [29], face recognition [30], object detection and localization [31], image characterization [32], etc. They are made up of neurons, where each neuron has a learnable weight and bias. It contains an **input layer**, an **output layer** and multiple hidden layers, where hidden layer consists of a **convolutional layer**, **pooling layer**, **fully connected layer (FC)** and various **normalization layers** [20] (See figure II.8 for ConvNet implementation in a similar network of LeNet).

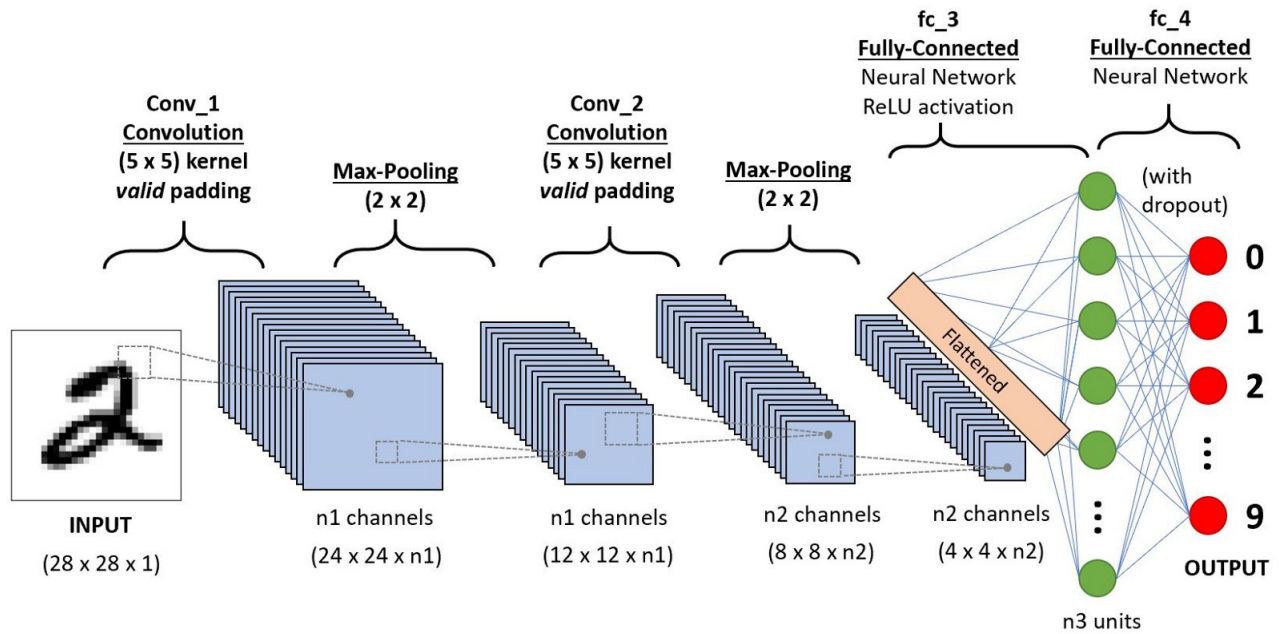


Figure II.8: Convolutional neural network [21]

Convolutional layer applies a convolution operation, to merge two sets of information. Pooling layer is used to reduce the dimensionality, by associating the output of neuron cluster at one layer with the single neuron. FC layer connects every neuron in one layer to every neuron in another layer. Its primary purpose is to classify the input images into several classes, based on the training datasets.

## II.6.2 Main components in a ConvNet

In a ConvNet we find the following components: input image, convolution layer, pooling layer and a fully connected layer [33].

### Input image

In the following figure we have a picture of the RGB which is divided into three color planes — red, green and blue. There are several color spaces that include images such as Grayscale, RGB, HSV, CMYK, and so on.

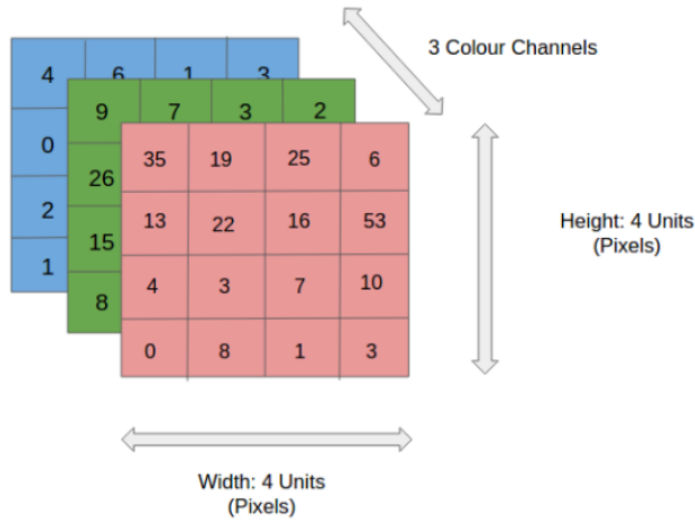


Figure II.9: Input image of size  $4 \times 4 \times 3$  [33]

ConvNet’s function is to transform the images, without losing the features that are necessary for good prediction, in a way that is easier to process. This is essential when designing an architecture that is not only useful for learning but can also be applied to large data sets.

**Convolution layer**

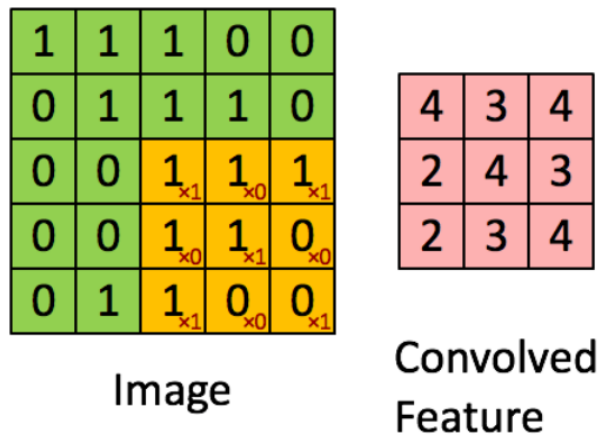


Figure II.10: Convoluting a  $5 \times 5 \times 1$  image with a  $3 \times 3 \times 1$  kernel to get  $3 \times 3 \times 1$  convolved features

[33]

In the above figure we have a green section which resembles the image of size  $5 \times 5 \times 1$ . The element that do the convolution operation is called the **kernel** (or **filter**) which is represented in yellow.

The kernel  $K$  used here is of size  $3 \times 3 \times 1$  and its value is:

$$K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (\text{II.10})$$

Each time a matrix multiplication operation is performed between  $K$  and the **portion**  $P$  of the image over which the kernel moves, the Kernel shifts 9 times by the **stride length**, which is 1 in this case. The figure II.11 shows how the kernel moves around an image.

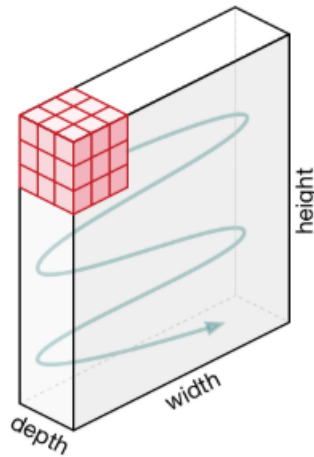


Figure II.11: Kernel movement  
[33]

The filter moves to the right with a certain stride value till it parses the complete width. Moving on, it goes down to the left side of the image with the same stride value and repeats the process until the entire image is traversed.

The convolution operation can be described by the following figure.



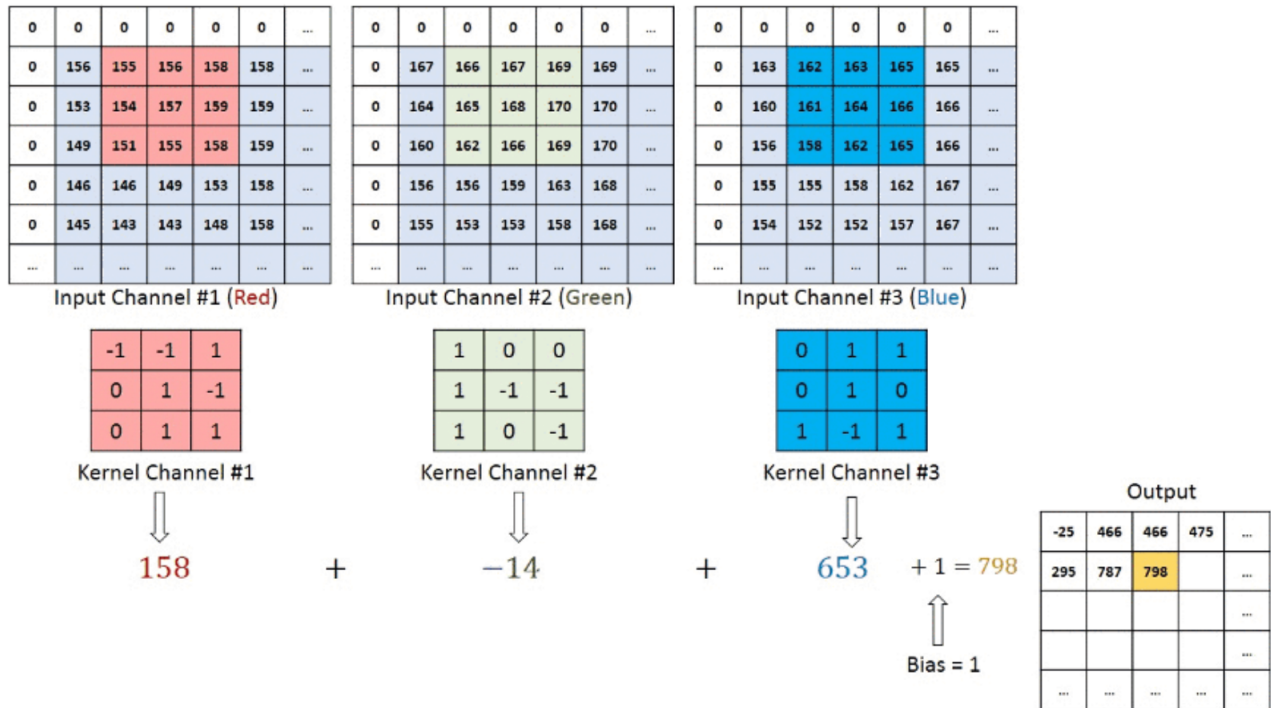


Figure II.12: Convolution operation on a  $M \times N \times 3$  image matrix with a  $3 \times 3 \times 3$  kernel [33]

This case is for images with depth field of 3, meaning there are 3 channels which are RGB. The kernel should have the same depth as the image, each channel is multiplied by a kernel for that channel and then we sum the values with bias to give the convoluted feature output.

The goal of the Convolution operation is to extract from the image of the input high-level elements such as edges [33]. ConvNets do not just have to be confined to one convolution layer. The first convolution layer usually takes care of the low-level features like edges, color, gradient orientation, etc. The architecture adapts with the additional layers to the high-level capabilities, giving us a network that understands images in the dataset in the same way as we do.

There are two types of results to the operation, one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **valid padding** in case of the former, or **same padding** in the case of the latter and that is all for a stride that equals to 1.

When we increase the image of  $5 \times 5 \times 1$  into an image of  $6 \times 6 \times 1$  and apply the kernel to the image, we discover that the matrix is  $5 \times 5 \times 1$  in size. Hence the name same padding.

On the other hand, if we perform the same operation without padding, we will get a matrix which has dimensions of the kernel ( $3 \times 3 \times 1$ ) itself, and that is valid padding.

### Pooling layer (Down-sampling)

As with the convolution layer, the Pooling layer reduces the convolved feature's spatial scale. This decreases the computing power required for data processing by reducing the dimension. In addition, it is useful for extracting dominant features that are invariant in rotation and position, so that the model is effectively trained.

Two forms of pooling are available: **max pooling** and **average pooling**. In Max Pooling, the maximum image section covered by the kernel returns the maximum value. The Average Pooling, on the other hand, returns the average value of all image values of the portion of a kernel image. The following figure shows the difference between them.

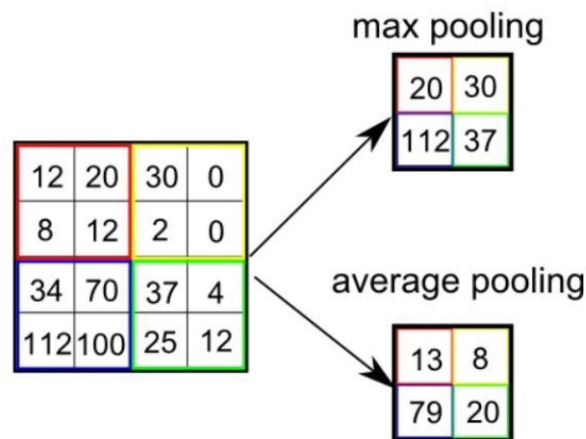


Figure II.13: Types of pooling  
[33]

Max Pooling is also a noise remover. It absolutely dismisses noisy activations and even denoises with reduced dimensionality. Average Pooling, on the other hand, actually decreases the dimensionality of noise. We may also tell that max pooling is much better than average pooling.

The convolution layer and the pooling layer together form the the first layer of the CNN. The number of layers can be increased to capture even more information of low levels, depending on the nuances in the images at the expense of more computing power.

Now the model can understand the features which defines the image. After this, the output of the first layer should be flattened to feed it into a regular neural network in order to train the model to get the results that we want.

### Fully connected layer

After converting our image to the appropriate form for our mutlilayer perceptron, we will flatten the image into a column vector (See figure II.14). The flattened output is provided for each training iteration by a feed forward neural network and back propagation. Over many periods, the model is capable, using the **Softmax classification technique**, of distinguishing between dominant and specific lower-level features of pictures.

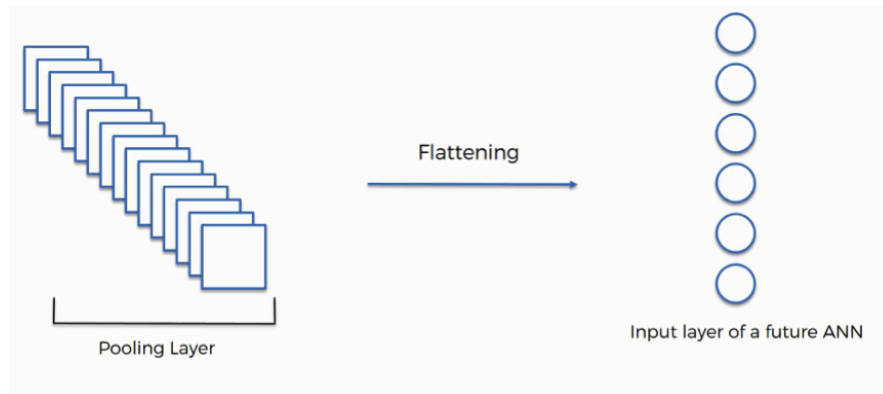


Figure II.14: Flattening  
[33]

## II.7 Conclusion

Artificial intelligence (AI) has many applications including machine learning (ML) which make the system learn from experience without specifying any explicit program. There are four methods in machine learning and those are: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. Many problems especially classification problems are using supervised learning algorithm where a model can be able to predict and classify based on what it was trained for. That model is represented as an artificial neural network (ANN) which is similar to an actual neural network in the human brain. From that, it was the initiation of deep learning (DL) with its architectures including convolutional neural network (ConvNet). The next chapter will be dedicated on the methodology of our project using ConvNets and some of the image processing techniques.

# Chapter III

## Methodology

## III.1 Introduction

To recognise road signs we present the used dataset which is the German traffic signs dataset (GTSD) that contains real images to help training our model, but before that we must go to some preprocessing stages in order to feed those images to the model so it can be more efficient especially when facing some disturbances like bad weather conditions.

In this chapter, we present an overview on the German traffic sign dataset, and then we will see some data augmentations techniques to have more images. After that, we present our approach of preprocessing stages in order to have effective predictions with even some disturbances. Finally, we will see the base ConvNet architecture and the fine-tuned architectures in the hope of obtaining a better model to recognise traffic signs.

## III.2 German Traffic Signs Dataset (GTSD)

This dataset contains real colored images of German traffic signs [19], more specifically it contains 43 classes for which each class represents a traffic sign. This dataset is smaller containing fewer images on training set, validation set and test set which means that each class have lesser amount of data, this can make a lot more difficult to train our network in order to obtain higher accuracy, but also it means that if we get a good accuracy, we can make it better when we add more data on new real images. The following figure shows the distribution of the train set.

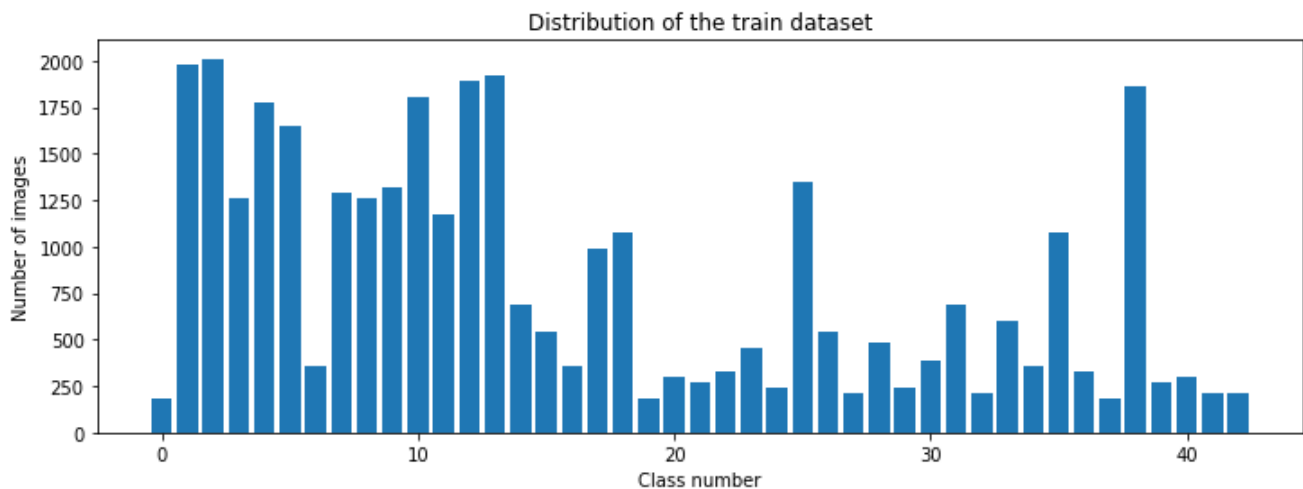
























Figure III.1: Distribution of the train set

As you can see from the figure above III.1, it is not well balanced which will make the training harder, and it may make the model biased to some classes which contains more data unlike the others which have a bit amount of data. This is a problem that can occur with real datasets, to solve this issue we should augment the data to have more data for each class so the model will be trained effectively. During the training, we can augment data using **data augmentation** technique which will make the training a bit longer but more efficiently.

The following table shows an overview on the GTSD dataset with all the classes with labels.

Table III.1: GTSDDB classes with labels

Class	Label	Example
0	Speed limit (20km/h)	
1	Speed limit (30km/h)	
2	Speed limit (50km/h)	
3	Speed limit (60km/h)	
4	Speed limit (70km/h)	
5	Speed limit (80km/h)	
6	End of speed limit (80km/h)	
7	Speed limit (100km/h)	
8	Speed limit (120km/h)	
9	No passing	
10	No passing for vehicles over 3.5 metric tons	
11	Right-of-way at the next intersection	
12	Priority road	
13	Yield	
14	Stop	
15	No vehicles	
16	Vehicles over 3.5 metric tons prohibited	
17	No entry	
18	General caution	
19	Dangerous curve to the left	
20	Dangerous curve to the right	
21	Double curve	

22	Bumpy road	
23	Slippery road	
24	Road narrows on the right	
25	Road work	
26	Traffic signals	
27	Pedestrians	
28	Children crossing	
29	Bicycles crossing	
30	Beware of ice/snow	
31	Wild animals crossing	
32	End of all speed and passing limits	
33	Turn right ahead	
34	Turn left ahead	
35	Ahead only	
36	Go straight or right	
37	Go straight or left	
38	Keep right	
39	Keep left	
40	Roundabout mandatory	
41	End of no passing	
42	End of no passing by vehicles over 3.5 metric tons	

From the table above we can also see the wide variety of images in each class, there are some images that are taken during night or even in different seasons like winter or summer which can absolutely help the model to be trained very well and to predict for other pictures in different conditions like bad weather conditions.

### III.3 Data augmentation

To face the problem of the lesser amount of data and the unbalanced number images of each class, we should use data augmentation in the training process to have new images so the model can be trained in various images of each class. In other words, the model can look at each image from a variety of different perspectives. Here's an example of data augmentation applied on one of the train images (III.2).



width\_shift\_range 0.1, height\_shift\_range 0.1, zoom\_range = 0.2, shear\_range = 0.1, rotation\_range = 10

Figure III.2: Sample of augmented data

As you can see from the figure above III.2, there is 5 augmented images for only one image from the dataset, which can help gain more data for our dataset during the training process.

This can help also make the model more effective in prediction because it can get more features for one image from the dataset by having more augmented images for just one image. Also these augmented images are not reducing the features at all, as a matter of fact it keeps the same features by having the same background in shifting operations, in other words the information it's not lost while the data augmentation process.

### III.4 Preprocessing stages

Since our dataset contains real images with highlighted backgrounds, this means that there are extra features which are unneeded to train our network and make it more difficult to classify them. Therefore, we can preprocess these images with two stages in order to make it easier for our network to classify them.

Here's an example applied on image from the GTS dataset which is represented in the following figure.



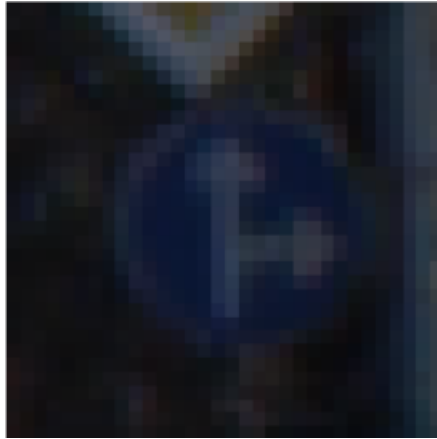


Figure III.3: Input image

Firstly, we convert colored images into grayscale images (See figure III.4 for the example above). This grayscale conversion is important for two main reasons:

- The first is that when distinguishing between traffic signs, color is not a very significant feature to look for since the lightning on our images varies and many of these traffic signs have similar colors.
- The features that are important to our classification problem are the edges, the curves, the shape inside of the signs, and that's what network should focus on. So, when we convert these images, we reduce the color depth from 3 to 1, and that means that our network will have fewer data since the input shape will have a depth of 1 channel, this also means that our network will be more efficient and require less computing time to classify our data.

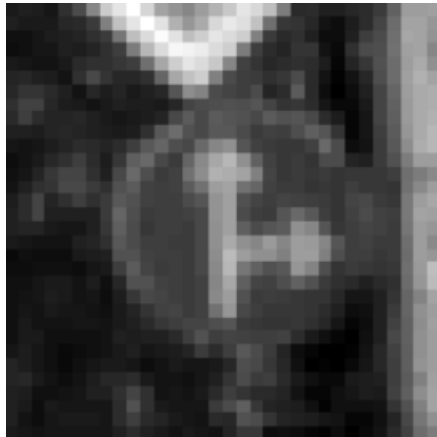


Figure III.4: Conversion to grayscale

Secondly, we do histogram equalization (See figure III.5 for the example above) in order to standardize the lightning in our grayscale images. Since all the images does not have the same lightning, after this stage we aim to have similar lightning effects to these grayscale images. This process also results in a higher contrast in our image which helps features extraction.

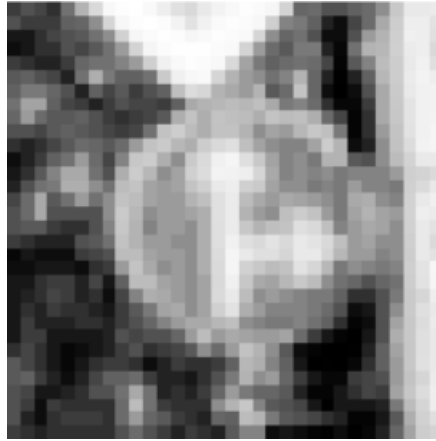


Figure III.5: Histogram equalization

### III.5 The proposed deep convnet architectures

In this section, we proposed to test LeNet model, which consist on two convolutional layers and two maxpooling, then two dense layers. The first one is followed by a dropout as the figure III.6 shows.

The reason of using LeNet as a base architecture is because of its simplicity and reliability for image classification, as well as its known high performance for other classification problems. Unlike other architectures which are deeper and more complex, LeNet can perform better with much less training time.

Also, LeNet model takes an image of size  $(32 \times 32)$ , which is the case with our German traffic signs dataset. That means also, after training and obtaining our model, in order to test it on real images we should resize them so we can preprocess them and feed them to our model.

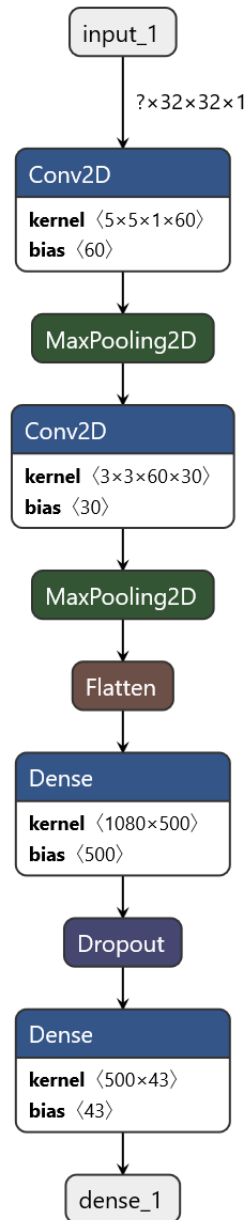


Figure III.6: LeNet base model

As you can see in the figure III.6, we have introduced a new layer which is called **dropout**. This special layer helps regularize DL networks and prevent the model to be overfitted by setting random nodes to zero during the training, this can helps other nodes (neurons) to learn well so the model can generalize the predictions.

Inspired from leNet base architecture, we proposed one other. Modified model 1 (See figure III.7), consist on two convolutional layers followed by a maxpooling layer, then two other convolutional layers also followed by a maxpooling layer, after that we add a dense layer with a dropout and finally another dense layer.

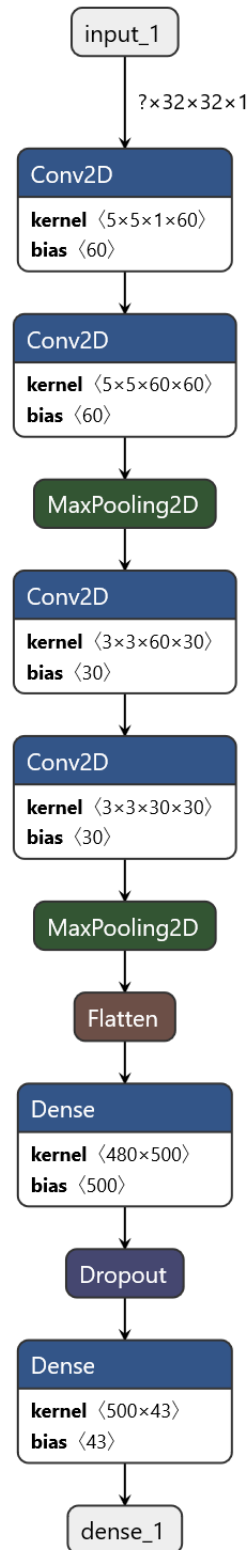


Figure III.7: Modified model 1 architecture

The second proposed model, noted modified model 2 is the same as modified model 1 before flattening, then two dense layers plus a dropout layer followed by a dense layer as the following figure III.8 describe.

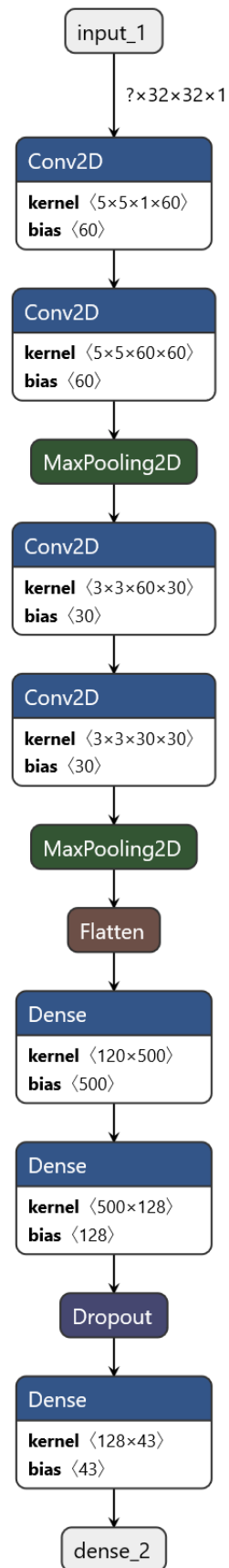


Figure III.8: Modified model 2 architecture

The reason of having two modified or fine tuned models is to compare the performances between them by having benchmark for the number of layers and even the optimizers used in training process.

In the following table we showed the number of parameters trained in each architecture. And also, the used optimizer, activation function and the loss correction.

Table III.2: Number of parameters in each used architecture

	<b>LeNet</b>	<b>Modified model 1</b>	<b>Modified model 2</b>
<b>Parameters</b>	579,833	378,023	246,155
<b>Activation function</b>	ReLU in hidden layers, Softmax in output layer		
<b>Loss correction</b>	Categorical_crossentropy		
<b>Optimizer</b>	Adam		RMSprop

We have fine-tuned the LeNet architecture by adding more convolutional layers to extract more features from each image and that can potentially improve the accuracy for our modified models. As we can see in the table III.2, the training parameters has been reduced for each modified model and that is because the image dimensions decrease after each convolutional layer also when add another dense layer in the FC layer, which affects the training process to have better performance.

Also we have used the same loss correction and the same activation functions but different optimizer between modified model 1 and modified model 2, and that is to see the impact of optimizer in training.

## III.6 Conclusion

This chapter was dedicated for our methodology to recognise traffic signs even with some disturbances that could cover some important features in the image. German traffic sign dataset was used to train the model because it's the most famous dataset for traffic signs in the world despite its unbalanced number of images in each class. If the model perform well in this dataset, that means it can perform even better with some custom dataset which contains many images with balanced number per each class. To face the problem of the unbalanced dataset, data augmentation technique has been used in order to gain more images so the model could be trained well. Moreover, image preprocessing technique was introduce by converting to grayscale and then apply image histogram for better lighting so the features can be more visible even for images for bad weather conditions. Finally, we introduced three ConvNet architectures to see which one of them is best after the training process, and that's what the following chapter is dedicated for, to discuss the train and test results, and to test the best model on new images as well as on images with bad weather conditions.

# Chapter IV

## Results & discussions

## IV.1 Introduction

After introducing our methodology for the classification problem of traffic signs, now we will see the results of the training process and to testing all three models on the test set in order to do benchmark between them to see which is more efficient and reliable to predict the classes. After that, using the best model we will test on new images from the web and then apply on them some artificial bad weather conditions to see if the model can still perform well.

## IV.2 Preliminary results

In this section, we first introduce the training results of dataset for the three models. The following table shows the number of images on each set. After that, we will select the best model to test on it and show the results.

Table IV.1: The distribution of GTS dataset

Sets	Number of images	Image shape
<b>Train set</b>	34799	(32, 32, 3)
<b>Test set</b>	12630	(32, 32, 3)
<b>Validation set</b>	4410	(32, 32, 3)

The results showed in the following are obtained after 100 epochs using batch size equal to 16 with a dynamic learning rate.

### IV.2.1 LeNet model

At first, for LeNet model we obtained 98.89% accuracy and 5% of loss. Then, we obtained 96.1% accuracy for the test set with 15% of loss.

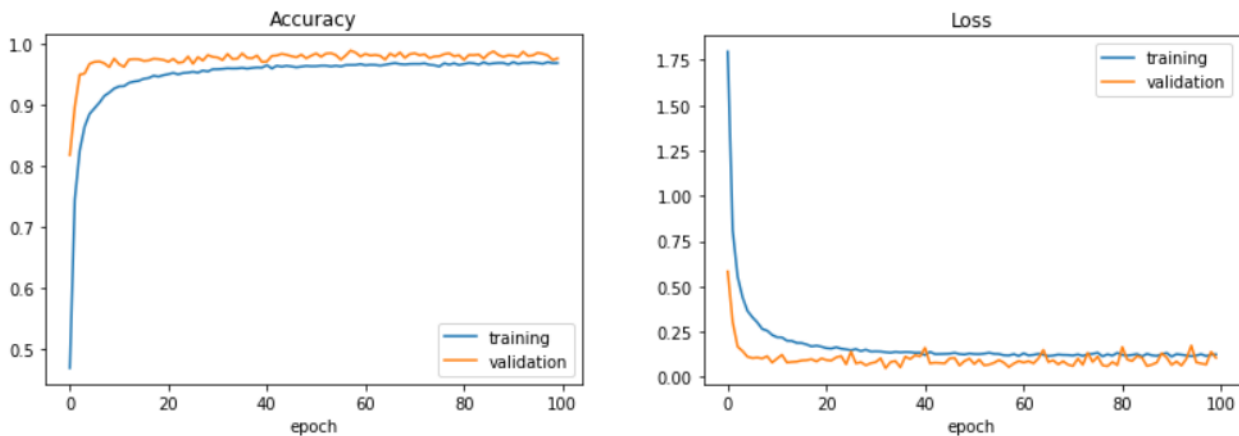


Figure IV.1: Results for LeNet model

The elapsed training time for this model was 2090718.49 ms, and the prediction time for one image was 36.57 ms.



## IV.2.2 Modified model 1

Secondly, for modified model 1, we obtained 99.43% accuracy and 3% of loss. Then, we obtained 97.21% accuracy for the test set with 13% of loss.

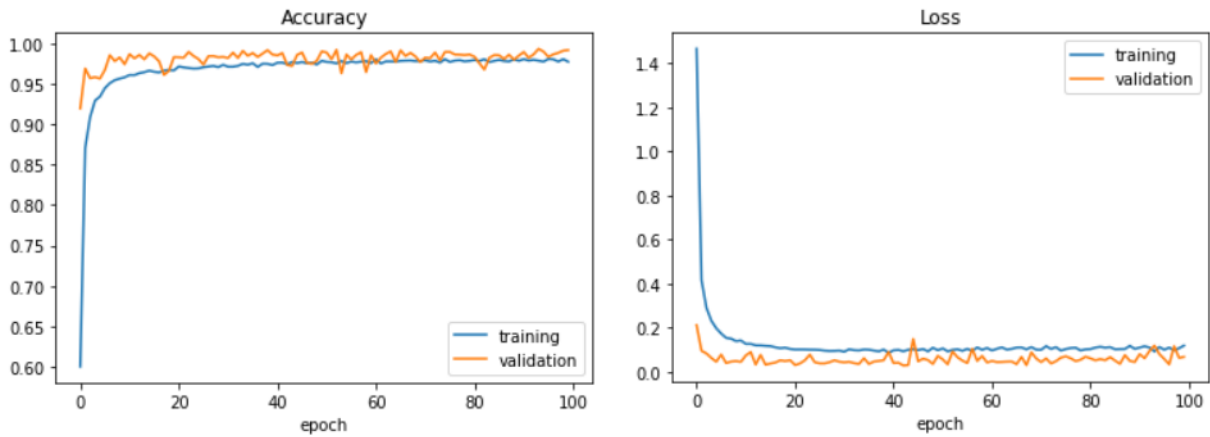


Figure IV.2: Results for modified model 1

The elapsed training time for this model was 2357516.47 ms, and the prediction time for one image was 31.17 ms.

## IV.2.3 Modified model 2

Finally, for modified model 2, we obtained 99.3% accuracy and 3% of loss. Then, we obtained 95.96% accuracy for the test set with 23% of loss.

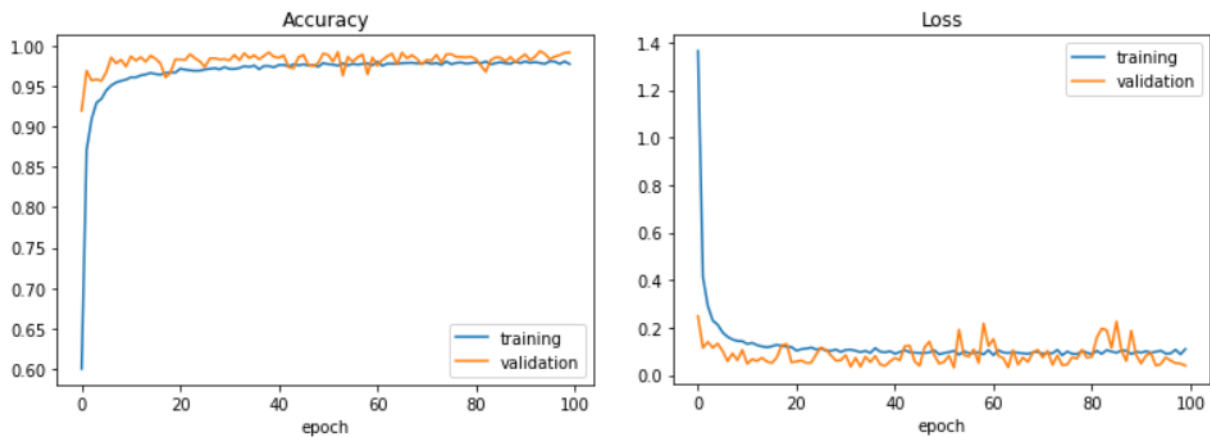


Figure IV.3: Results for modified model 2

The elapsed training time for this model was 2344391.19 ms, and the prediction time for one image was 31.40 ms.

The following table IV.2 shows a summary of all results we had for the three models.

Table IV.2: Preliminary results for all models

Models	LeNet	Modified model 1	Modified model 2
<b>Validation accuracy</b>	98.89 %	99.43 %	99.3 %
<b>Validation loss</b>	5 %	3 %	3 %
<b>Test accuracy</b>	96.1 %	97.21 %	95.96 %
<b>Test loss</b>	15 %	13 %	23 %
<b>Training time</b>	2090718.49 ms	2357516.47 ms	2344391.19 ms
<b>Prediction time on one image</b>	36.57 ms	31.17 ms	31.40 ms

These results are obtained from the best models for each architecture from each epoch and that is by taking the model that has the lesser validation loss, because in reality we want the model that can predict on many pictures, in other words the validation loss defines which is the best model.

#### IV.2.4 Discussion

From the results above, we can conclude that the best model is the modified model 1 with 98% accuracy and 8% of loss in the test set. As the figures above shows, the three models can generalize predictions on new images since the validation accuracy is higher than train accuracy, the same applies to the validation loss which is lower than the training loss, in other words all three models are not overfitting. But the modified model 1 showed better accuracies. Also, we can say than the use of Adam optimizer is slightly better than RMSprop because from the results we've seen even when we changed the architectures a bit (LeNet and modified model 1) there is a similar curve for the training set and it guarantees to find a better parameters corresponding to the minima, however, when we changed the optimizer for another architecture (modified model 2) the curve for the training has been changed in a significant way and shows how it struggles to find the best parameters and sometimes it's even overfitted when the validation loss is greater than the training loss. At the end, deep convolutional layer (4 layers) with two dense layers is sufficient for give good results, no need to go deeper than that.

### IV.3 Testing on new images

Using the best model which is modified model 1, and evaluating it on the test set which contains real images that were not seen by the model during the training, we have obtained the following confusion matrix (See figure IV.4) which shows the corrected labels with predicted ones to show how this model can perform better on new images.

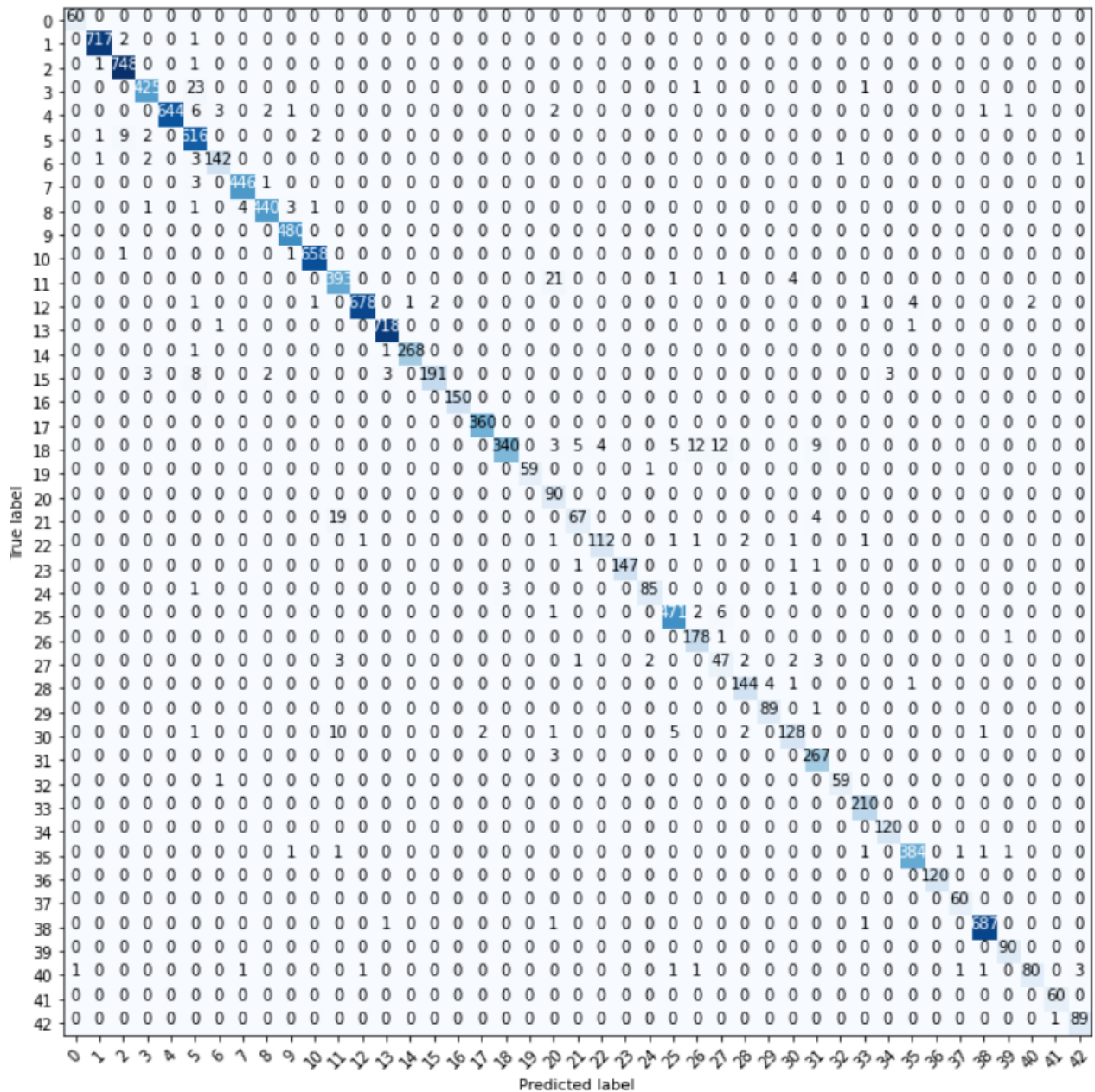


Figure IV.4: Confusion matrix

### IV.3.1 Discussion

At first glimpse, we see that there are many elements (images) on the diagonal which is good because it means the model predicted correctly the new images. However, there are some false predictions which are the elements other than the diagonal and that is because of the unbalanced training set of GTSD.

But in whole view we can see that this model has done very well by predicting correctly even if it was trained on a unbalanced dataset which means it can perform better on another big and balanced dataset.

We also test to recognize a stop panel (See figure IV.5), the image is downloaded from the web and it's not included in the dataset, prediction is done perfectly with 98.64% of accuracy.

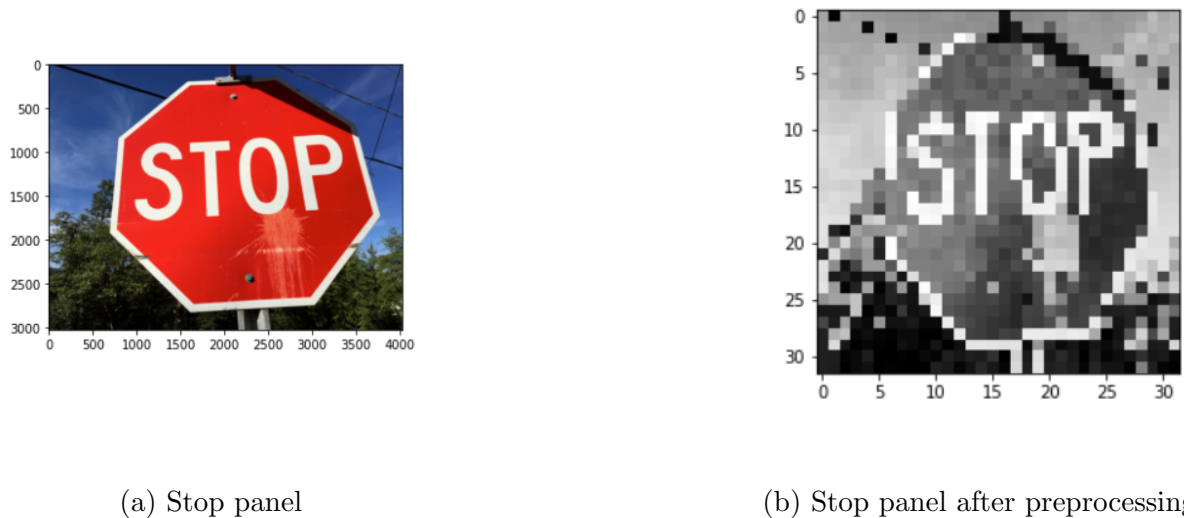


Figure IV.5: Prediction on a stop panel image

As the figure IV.5 shows, the stop panel IV.5a has a size of  $(3000 \times 4000)$ . However in order to feed it to our model it should be of size  $(32 \times 32)$  after the preprocessing stages (See sub-figure IV.5b) because it's how the model was trained.

Likewise we have tested our model to another stop panel image with size  $(300 \times 300)$  as the following figure shows. The prediction was great with 99.99 % of accuracy.

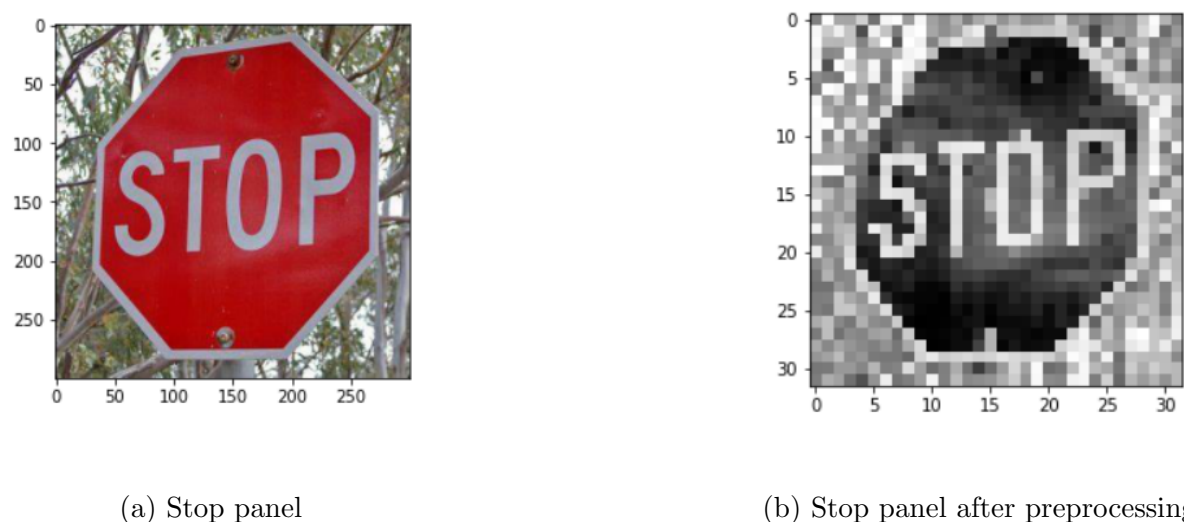


Figure IV.6: Prediction on another stop panel image

From this prediction we conclude that the less size of the image is the better, because when

we reduce the size of a small image the information will not be lost as much as for a bigger image.

## IV.4 Testing on images with bad weather conditions

In this section we will test our model on images with bad weather conditions to see if our technique of preprocessing is working.

In order to do that we come up with new images from the web and then we applied some artificial rain and snow and even fog to it, each time we add that disturbance to the image we predict and see if the model is still predicting well.

So at first we started with a slippery road sign and the add to it those artificial disturbances as the figure IV.7 shows.

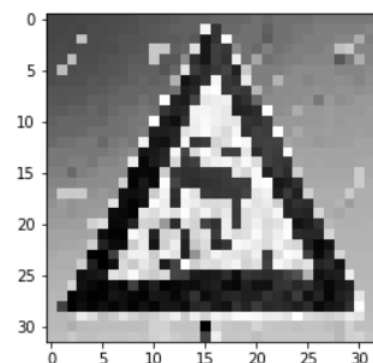
The following table IV.3 shows how many iterations we did of adding artificial disturbances with the predictions each time.

Table IV.3: Predictions on the slippery road sign with bad weather conditions

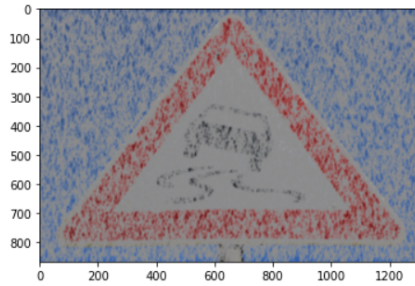
Noise round	Accuracy	Prediction
0	100 %	correct
1	100 %	correct
2	100 %	correct
3	100 %	correct
4	100 %	correct
5	100 %	correct
6	99.99 %	correct
7	99.55 %	correct
8	55.34 %	false



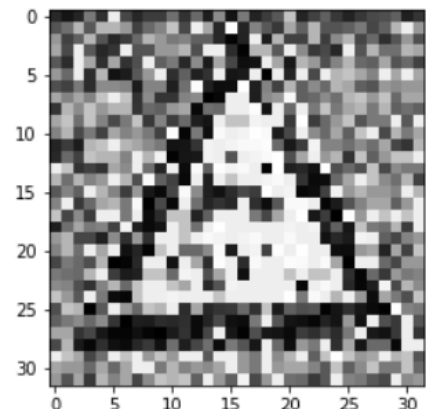
(a) Slippery road with no disturbance (noise round 0)



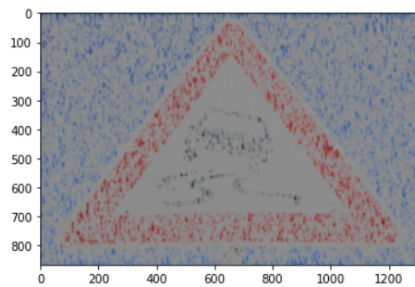
(b) Slippery road with no disturbance after preprocessing (noise round 0)



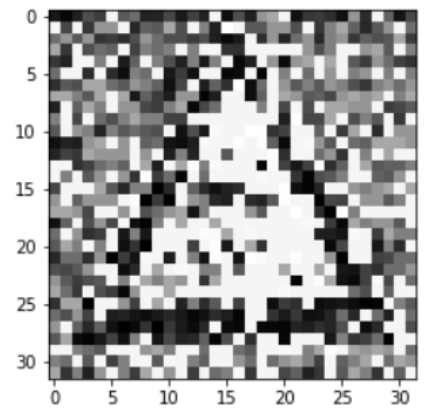
(c) Slippery road with some disturbance  
(noise round 6)



(d) Slippery road with some disturbance af-  
ter preprocessing (noise round 6)



(e) Slippery road with more disturbance  
(noise round 8)



(f) Slippery road with more disturbance af-  
ter preprocessing (noise round 8)

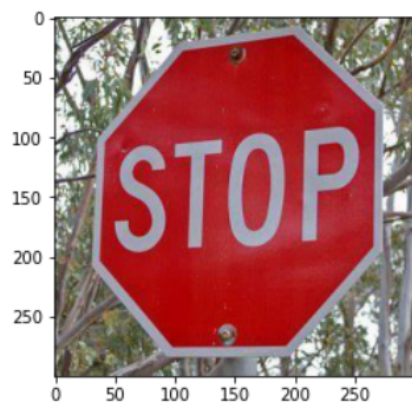
Figure IV.7: Prediction on slippery road sign with bad weather conditions

For comparison we did the same thing for another panel sign which is the stop panel (See figure IV.8). However, in this case in each iterations we have added more disturbances like thick rain drops and more fog to it. The following table IV.4 shows the result for the predictions of stop sign with those disturbances.

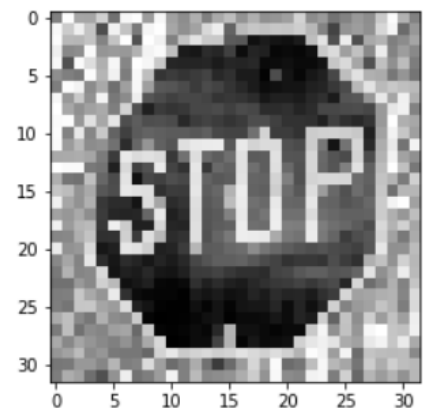
Table IV.4: Prediction on stop sign with bad weather conditions

Noise round	Accuracy	Prediction
0	100 %	correct
1	100 %	correct
2	99.99 %	correct
3	4.50 %	false

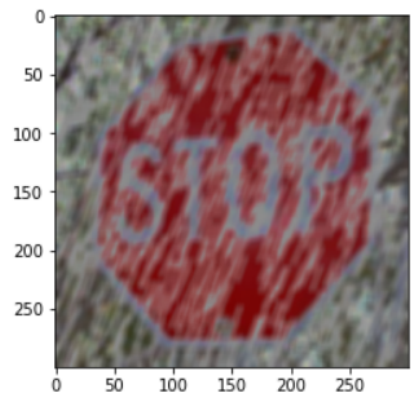




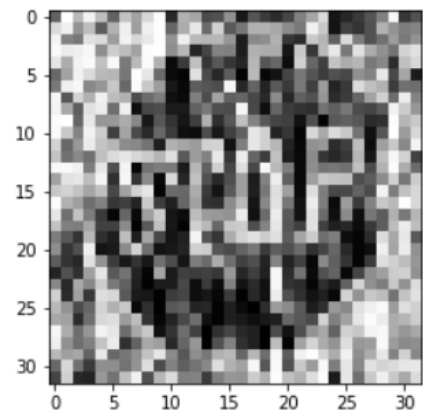
(a) Stop sign with no disturbance (noise round 0)



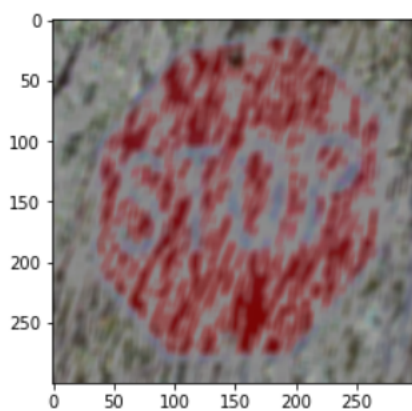
(b) Stop sign with no disturbance after pre-processing (noise round 0)



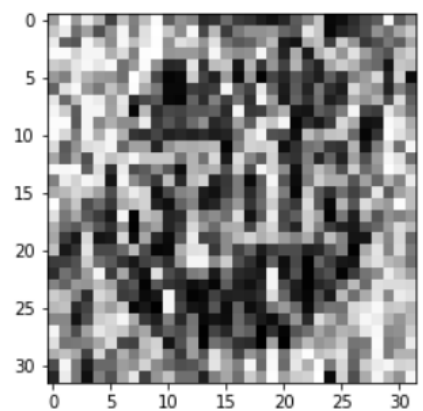
(c) Stop sign with some disturbance (noise round 2)



(d) Stop sign with some disturbance after preprocessing (noise round 2)



(e) Stop sign with more disturbance (noise round 3)



(f) Stop sign with more disturbance after preprocessing (noise round 3)

Figure IV.8: Prediction on stop sign with bad weather conditions

### IV.4.1 Discussion

From the results above we can understand that the noises can indeed affect the predictions. For the first attempt with slippery road sign we saw that the model kept predicting correctly after 7 noise rounds, however for the second attempt for the stop sign it last for 2 noise rounds which means how the disturbances can affect the classifying by covering some necessary features of the image. However, we see that our model has performed very well on classifying road signs in bad weather conditions after the preprocessing stages, because that is the model input, and we can also say even for a human being it's hard to classify those images with disturbances after preprocessing. Also, the resizing can also affect the prediction because some information of the image can be lost which means some features will be lost.

## IV.5 Conclusion

This chapter was devoted to discuss the preliminary results of the training and testing process for all three architectures. From that, we picked the best model on the criteria of which is has the lesser validation loss that was 2 %, the validation accuracy was also pretty good which was 99.39 %. Using the best model we tested it on new images to see if it's still performing well and the results were just fine, even with some artificial disturbances which resembles bad weather conditions. Finally, we conclude that we indeed create an automatic recognition system for road signs and it also performed well with road signs in bad weather conditions.



# General conclusion

In this project, we were interested to face the problem of classifying road signs in bad weather conditions because many accidents occurs as a result of poorly visible traffic panels from natural effects like rain, fog or even snow. From that initiative we present three deep convolutional neural networks for traffic sign recognition. The first model is LeNet model widely used for image classification. Then the two other models are LeNet modified on more deeper models. We evaluated them under different circumstances and hyper parameters to properly tuning the proposed models.

To realize this project, we have started by seeing the state of art to see where the current solutions has arrived. After that, we talked about theoretical notions for machine learning and deep learning and in particularly artificial neural networks and ConvNets. Then we introduced our methodology to solve the problem of classification where we show some techniques that have been used to face the problem of the unbalanced German traffic signs dataset and to contribute our work to classify in bad weather conditions. Lastly, the results have shown how efficient our automatic recognition system for traffic signs in normal and bad weather conditions.

Another interesting aspect of this work that can be explored in the future would be to test this approach on more databases with night-time vision, and to add detection step by using one of the famous detection architectures and the result of that detection can be applied to our model as to recognize road signs very effectively, as it can also applied for video sequences. This application will be useful for an objective driving aid to contribute in reducing road accidents caused by poor visibility.

A part of this work has been accepted as a research paper in the 4th International Workshop on Connected and Intelligent Mobility (CIM 2020) [34].

# Bibliography

- [1] “Accidents de la route.” <https://www.who.int/fr/news-room/fact-sheets/detail/road-traffic-injuries> (accessed Apr. 18, 2020).
- [2] P. Mikoski, G. Zlupko, and D. A. Owens, “Drivers’ assessments of the risks of distraction, poor visibility at night, and safety-related behaviors of themselves and other drivers,” *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 62, pp. 416–434, Apr. 2019.
- [3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
- [4] F. Sultana, A. Sufian, and P. Dutta, “Advancements in Image Classification using Convolutional Neural Network,” *ArXiv190503288 Cs*, May 2019.
- [5] “MNIST Demos on Yann LeCun’s website.” <http://yann.lecun.com/exdb/lenet/> (accessed Apr. 21, 2020).
- [6] S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad, “An automatic traffic sign detection and recognition system based on colour segmentation, shape matching, and svm,” *Math. Probl. Eng.*, vol. 2015, 2015.
- [7] S.-C. Huang, H.-Y. Lin, and C.-C. Chang, “An in-car camera system for traffic sign detection and recognition,” in *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, pp. 1–6, Jun. 2017.
- [8] A. Ellahyani, M. El Ansari, and I. El Jaafari, “Traffic sign detection and recognition based on random forests,” *Appl. Soft Comput.*, vol. 46, pp. 805–815, 2016.
- [9] Z. Abedin, P. Dhar, M. K. Hossenand, and K. Deb, “Traffic sign detection and recognition using fuzzy segmentation approach and artificial neural network classifier respectively,” in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 518–523, 2017.
- [10] “Image (color?) segmentation with opencv C++,” *Stack Overflow*. <https://stackoverflow.com/questions/30303151/image-color-segmentation-with-opencv-c> (accessed Sep. 11, 2020).
- [11] K. Islam and R. Raj, “Real-time (vision-based) road sign recognition using an artificial neural network,” *Sensors*, vol. 17, no. 4, p. 853, 2017.

- [12] S. Khalid, N. Muhammad, and M. Sharif, "Automatic measurement of the traffic sign with digital segmentation and recognition," *IET Intell. Transp. Syst.*, vol. 13, no. 2, pp. 269–279, 2018.
- [13] S. Pei, F. Tang, Y. Ji, J. Fan, and Z. Ning, "Localized traffic sign detection with multi-scale deconvolution networks," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 355–360, 2018.
- [14] X. Liu, Z. Deng, and Y. Yang, "Recent progress in semantic image segmentation," *Artif Intell Rev*, vol. 52, no. 2, pp. 1089–1106, Aug. 2019.
- [15] A. Jose, H. Thodupunoori, and B. B. Nair, "A Novel Traffic Sign Recognition System Combining Viola–Jones Framework and Deep Learning," in *Soft Computing and Signal Processing*, Springer, pp. 507–517, 2019.
- [16] A. Mannan, K. Javed, S. K. Noon, and H. A. Babri, "Optimized segmentation and multiscale emphasized feature extraction for traffic sign detection and recognition," *J. Intell. Fuzzy Syst.*, no. Preprint, pp. 1–16, 2019.
- [17] E. S. Team, "What is Machine Learning? A definition," *Expert System*, Mar. 2017.
- [18] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.
- [19] "German Traffic Sign Benchmarks." <http://benchmark.ini.rub.de/?section=gtsdb&subsection=dataset> (accessed Apr. 21, 2020).
- [20] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Prog Artif Intell*, Dec. 2019.
- [21] A. Sharma and D. Kumar, "Classification with 2-D Convolutional Neural Networks for breast cancer diagnosis", Jul. 2020.
- [22] X.J. Wang, L.L. Zhao, S. Wang, "A novel SVM video object extraction technology". 8th International Conference on Natural Computation, IEEE, pp. 44–48, 2012.
- [23] I. Rish, "An empirical study of the naive Bayes classifier", *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22, pp. 41–46, 2001.
- [24] N. Islam, I. Zeeshan, N. Nazia, "A survey on optical character recognition system", arXiv preprint arXiv:1710.05703, 2017.
- [25] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. WardeFarley, S. Ozair, A.C. Courville, Y. Bengio, "Generative adversarial networks", arXiv:1406.2661, 2014.
- [26] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biol. Cybern.* 36, 193–202, 1980.
- [27] N. Yudistira, T. Kurita, "Gated spatio and temporal convolutional neural network for activity recognition: towards gated multimodal deep learning", *EURASIP J. Image Video Process*, 2017.

- [28] Y. Kim, "Convolutional neural networks for sentence classification", arXiv preprint arXiv:1408.5882, 2011.
- [29] X. Zhou, W. Gong, W. Fu, F. Du, "Application of deep learning in object detection", ACIS 16th International Conference on Computer and Information Science (ICIS), pp. 631–634. IEEE, 2017.
- [30] R. Ranjan, S. Sankaranarayanan, A. Bansal, N. Bodla, J. C. Chen, V. M. Patel, C.D. Castillo, R. Chellappa, "Deep learning for understanding faces: machines may be just as good, or better, than humans", IEEE Signal Process. Mag. 35(1), 66–83, 2018.
- [31] S. Milyaev, I. Laptev, "Towards reliable object detection in noisy images", Pattern Recognit. Image Anal. 27(4), 713–722, 2017.
- [32] P. N. Druzhkov, V. D. Kustikova, "A survey of deep learning methods and software tools for image classification and object detection", Pattern Recognit. Image Anal. 26(1), 9–15, 2016.
- [33] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," Medium, Dec. 17, 2018. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (accessed Sep. 01, 2020).
- [34] O. Belghaouti, W. Handouzi, M. Tabaa, "Improved Traffic Sign Recognition Using Deep ConvNet Architecture", The 4th International Workshop on Connected and Intelligent Mobility (CIM 2020), accepted.

## ملخص

من أجل تحسين السلامة على الطرق وتقليل مخاطر الحوادث بشكل كبير بسبب ضعف إشارات المرور بسبب سوء الأحوال الجوية ، تم تنفيذ العديد من الأعمال. في هذا العمل ، نقترح مساهمة في هذه المشكلة من خلال الاستفادة من النتائج الرائعة للشبكة العصبية التلافيفية العميقة في رؤية الكمبيوتر. نقترح نظام التعرف التلقائي على إشارات الطريق بناءً على نموذج معدل مستوحى من نموذج لينيت. النتائج التي تم الحصول عليها من خلال مقارنة نموذج لينيت ونموذجين معدلين مقترحين على مجموعة بيانات حركة المرور الألمانية تبلغ دقتها حوالي 99% وهو أمر واعد مقارنة بأحدث النتائج المتعلقة بتصنيف إشارات المرور في الظروف الجوية السيئة.

**الكلمات الدالة :** نظام التعرف التلقائي على إشارات الطريق ، شبكة العصبية التلافيفية ، مجموعة بيانات حركة المرور الألمانية ، لينيت .

## Résumé

Afin d'améliorer la sécurité routière et de réduire significativement les risques d'accidents dus à une mauvaise visibilité des panneaux routiers en raison de mauvaises conditions météorologiques, de nombreux travaux ont été entrepris. Dans ce travail, nous proposons une contribution à ce domaine en tirant parti des résultats remarquables des réseaux neuronaux convolutionnels (ConvNet) profonds en vision par ordinateur. Nous proposons un système de reconnaissance automatique des panneaux de signalisation basé sur un modèle modifié inspiré de l'architecture LeNet. Les résultats obtenus par comparaison du modèle LeNet et des deux modèles modifiés proposés sur la base de données (GTSD) sont d'une précision d'environ 99 %, ce qui est prometteur par rapport aux résultats obtenus dans la revue bibliographique. Ces modèles ont également donné de bons résultats pour la classification des panneaux dans de mauvaises conditions météorologiques.

**Mots clés :** Reconnaissance des panneaux de signalisation, ConvNet, base de données des panneaux de signalisation allemands (GTSD), LeNet.

## Abstract

In order to improve road safety and significantly reduce the risk of accidents due to poorly visible road signs because of bad weather conditions, numerous works have been undertaken. In this work, we propose a contribution to this issue by taking advantage of the remarkable results of deep ConvNet in computer vision. We propose an automatic recognition system of road signs based on a modified model inspired by LeNet model. The results obtained by comparison of LeNet model and two proposed modified models on the German traffic dataset is about 99 % accuracy which is promising compared to the state-of-the-art results which also showed promising results on classifying traffic signs with bad weather conditions.

**Keywords :** Traffic sign recognition, ConvNet, German Traffic Signs Dataset (GTSD), LeNet.