DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

TRAFFIC SIGN RECOGNITION WITH MACHINE LEARNING METHODS

by

Emin Alper SÜRÜCÜ

October, 2019

İZMİR

TRAFFIC SIGN RECOGNITION WITH MACHINE LEARNING METHODS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences Dokuz Eylül University
In Partial Fullfilment of the Requirements for the Degree of Master of Science in Electrical and Electronics Engineering

by

Emin Alper SÜRÜCÜ

October, 2019

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"TRAFFIC SIGN RECIGNITION WITH MACHINE LEARNING METHODS"** completed by **EMİN ALPER SÜRÜCÜ** under supervision of **ASST. PROF. HATİCE DOĞAN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Hatice DOĞAN

Supervisor

Assoc. Prof. Dr. Güleser K. Demir

(Jury Member)

Asst. Prof. Dr. Göktan Demirkaran

(Jury Member)

Prof.Dr. Kadriye ERTEKİN

Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Asst. Prof. Hatice Doğan of the Electric-Electronic Engineering at Dokuz Eylül University for always pushing me on and her precious knowledge and patience.

I am grateful because Tuğçe Toprak and Özge Canlı were encouraging me every time to finish my thesis. I appreciate that effort.

I also thank for my family for supporting me all the time. It wouldn't have been a complete study without them.

Emin Alper SÜRÜCÜ

TRAFFIC SIGN RECOGNITION
WITH MACHINE LEARNING METHODS

ABSTRACT

Traffic sign recognition is an important subject for the vehicles that are autonomously controlled. Since there is no human intervention in this type of vehicle, all information about the environment is collected by means of sensors and systems such as camera, distance sensor, RFID, sonar sensor and GPS. One of the important requirements for autonomous vehicles is the recognition of traffic signs. In the first methods proposed for feature extraction in the identification of traffic signs was generally performed by computer vision method, and classification was performed by using these attributes, but due to the large and fast increase in GPU performance, feature extraction was done by machine learning methods.

CNN has become one of the most used deep learning method thanks to its state-of-the art performance at tough problems. With CNN, superior success has been achieved in traffic sign recognition which is crucial for autonomous vehicles. In this study, two-stage hierarchical CNN structure is proposed in order to classify traffic signs. Signs are divided into 4 main groups at the first stage by using similarity index. And then classes of each main group are subclassified with CNNs at the second stage. Performance of the network is examined on 43-classes GTSRB dataset, compared with structures using different similarity index and methods proposed in thesis .

Keywords: CNN, hierarchical, traffic signs, deep learning, GTSRB

MAKİNE İLE ÖĞRENME YÖNTEMLERİYLE TRAFİK İŞARETLERİ TANIMA

ÖZ

Trafik işareti tanıma otonom kontrol edilebilen araçlar için önemli bir konudur. Bu tip araçlara herhangi bir insan müdahalesi olmadığı için çevre hakkındaki bütün bilgiler kamera, uzaklık sensörü, RFID, sonar gibi sensörler ve GPS gibi sistemler aracılığıyla toplanır. Otonom araçlar için önemli ihtiyaçlardan biri de trafik işaretlerinin tanınmasıdır. Trafik işaretlerinin tanınması probleminin çözümü için önerilen ilk yöntemlerde, işaretlerin sınıflandırılması için gerekli öznitelikler görüntü işleme yöntemi ile çıkarılır ve sınıflandırma bu özniteliklerden faydalanılarak yapılırdı. Grafik İşlemci Birimlerinin (GPU) performanslarındaki büyük ve hızlı artışın etkisiyle öznitelik çıkarımı için derin öğrenme yöntemleri kullanılmaya başlanmıştır.

KSA zorlu problemlerin çözümünde gösterdiği üstün performans sayesinde son yıllarda en çok kullanılan derin öğrenme yöntemlerinden biri olmuştur. Sürücüsüz araçlar için kritik olan trafik işaretlerinin tanınması probleminde de KSA ile yüksek başarımlar elde edilmiştir. Bu çalışmada trafik işaretlerinin sınıflara ayrılabilmesi için iki katmanlı hiyerarşik KSA yapısı önerilmiştir. Yapısal benzerlik indisi kullanılarak tasarlanan yapının ilk katmanında işaretler 4 ana gruba ayrılmıştır. Ardndan yapının ikinci katmanında ise her ana gruptaki işaretler KSA'lar ile alt sınıflara ayrılmıştır. Yapının performansı 43 sınıftan oluşan GTSRB veri seti üzerinde hesaplanmış, farklı benzerlik indisi kullanan hiyerarşik yapılar ve yazında önerilen diğer yöntemlerle karşılaştırılmıştır.

Anahtar Kelimeler: KSA, hiyerarşik, trafik işaretleri, derin öğrenme, GTSRB

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

Nowadays, the most of the companies that are aiming to make peoples life much more easier are using Artificial Intelligence (AI). Automobile companies, banks, firms that are on social media, mobile phones applications, smart home systems are all taking advantage of AI. Recently, considerable amount of people are working on developing the actual systems with AI so as to give better performance beyond the state-of-art structures. One of the application areas of AI is the autonomous driving systems. There are rules to ensure safety in traffic and to reduce accident risks. Most of these rules are shown in traffic signs because this is the simplest and easiest way to inform drivers. Autonomous driving system must have the ability to recognize traffic signs. The critical parts of this system are how the informations about environment will be collected and how these informations will be processed in order to get a meaningful inference. The first part needs the necessary tools for this purpose like sensor, GPS and these are all about hardware phase of the system. The second part which is evaluating these data is mostly related with software phase. For autonomous vehicles, traffic signs are so crucial so that the vehicle decides how to act according to traffic signs and other informations evaluated by the system. One of the hardest part of this system is the recognition of traffic signs from the images captured by the camera mounted on the vehicle. The captured images are effected from illumination conditions, orientation, speed variations of car etc. Machine learning provides different solutions to this problem.

Before spreading of deep learning methods, traffic sign recognition can be done by extraction of the features from the images by image processing algorithms (Hasasneh et al., 2016; Zhu et al., 2016; Møgelmose et al., 2012; Li et al., 2015; Hossain et al., 2015) but the accuracies of these systems weren't at the level what companies and researches desired therefore these methods hadn't been used in such a systems where the life of people matters. After having great experience about and accepting the ability of deep learning, it got used to be indispensable As an example of this evolving, there is a competition to held every year in visual recognition called 'ImageNet Challenge' and

this challenge consists of lots of categories as target inputs to be recognized accurately. It's not a traffic sign recognition competition and the categories are changed for every year. It could be any classification. Before 2012, almost all the participants had been using classical hand-crafted methods but the state-of-the-art performances weren't satisfying . In 2012, a group of researchers from University of Toronto (Krizhevsky et al., 2012) took the record with 15.2 % error rate for test set using CNN while the second best performance in error rate was 26 %. It was the milestone for this visual recognition challenge and after 2012, most of the participants tried to achieve the best performance by taking advantage of deep learning methods. With the understanding that the methods of deep learning are superior to other classical methods, it has brought demand for deep learning and its algorithms. CNN, the structure just mentioned, is a method of deep learning that has become quite popular recently. Deep learning is a subgroup of machine learning as shown in Figure 1.1. The methods used before deep learning takes place in general by making use of the features given to the system, but in deep learning, features are also learned by the network. Info about machine learning and its one of the most important methods will be given in next chapter.

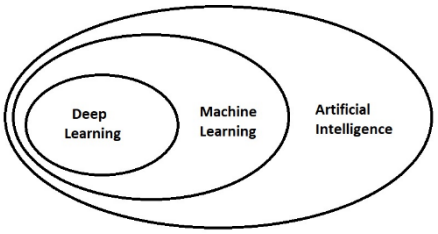


Figure 1.1 Hierarchy of deep learning, machine learning and artificial intelligence

The convolutional neural networks (CNN) which is popular technique of deep learning for the analysis of visual imagery are also used for the traffic sign recognition and their performances are outstanding (Ciregan et al., 2012; Qian et al., 2016; Stallkamp et al., 2012) compared to the standard approaches (Haloi, 2015b; Yin et al., 2015).

In this thesis, a hierarchical CNN structure is proposed for the recognition of traffic signs. The aim of the hierarchical structure is to categorize the classes into groups so that the classes in one group are similar to each other than the classes of the other group

then each groups are classified individually. Hierarchical structures are inspired from human coarse to fine learning approach (Mao et al., 2016). The main advantage is that a smaller network with less classes is easier to learn than a large network (Mao et al., 2016). Proposed hierarchical structures are build using two different similarity methods and they are employed on German Traffic Sign Recognition Benchmark (GTSRB) and performances are compared with the other methods in literature. The rest of the thesis is organized as follows: the related works are summarized in Chapter 2, methodology is explained in Chapter 3, details of the proposed methods are given Chapter 4, results are presented in Chapter 5 and conclusions are mentioned in Chapter 6.

CHAPTER TWO
RELATED WORKS

Traffic sign recognition is a constantly evolving area due to the growing interest in autonomous driving research. Different methods are proposed in the literature for the detection and recognition of the traffic signs that can be classified into three main groups as color based, shape based and learning based methods. In color and shape-based methods the expected color and shape of a traffic sign is manually encoded into the solution (Brkic, 2010).

In learning based approaches machine learning methods are used. CNN is the most popular one and the proposed method of thesis is done with CNN but the first researches will be examined are not relied on CNN. The first one is hierarchical sign recognition (Yunxiang Ma, 2015). This work is done by the combination of extracting histogram of gradient (HoG) and support vector machines (SVM) and the recognition rate is 98.76 %. As a hand-crafted method for traffic sign recognition, there is small amount of work aiming to develop an algorithm for classification of Italian de-restriction signs which has 33 of 37 accuracy rate (Caraffi et al., 2008). In (Zakir et al., 2010), road signs are classified in 6 main colors and has accuracy of 90-95 %.

There are lots of papers published not only for traffic sign recognition but also any area in which the method is CNN. For example H. Shirani-Mehr worked on sentiment analysis by using a different type of network and made comparison between Naive Bayes, Recurrent Neural Network and Convolutional Neural Network (Shirani, 2015). The performances aren't as high as any performance that can be obtained in any visual study using CNN. Another topic not related to traffic sign subject is network traffic identification in which the features in the flow data can't be extracted easily. So Z. Wang applied deep learning on this topic and obtained considerable result (Wang, 2015).

As deep learning proved its success in image analysis, in the literature different versions of deep learning methods have been proposed to solve traffic sign recognition

problem. In this part the studies will be related to the traffic sign recognition done with CNN. The one that reaches 99.81 % accuracy is a deep network consisting of spatial transformer layers and modified version of inception module (Haloi, 2015a) designed to capture local and global features together. There is another study (Youssef et al., 2016) using CNN which has two parts including detection and recognition. For detection, some pre-processing are applied to the system and for the recognition multi-scale CNN is used. Linear discriminant analysis and convolutional neural network is compared in the study (Bodington et al., 2015) and it has been obtained that recognition rates are 98.25 % for LDA and 98.75 % for CNN respectively when classifying eight different types of traffic signs. The CNN was implemented on a GPU for real-time traffic sign classification. Another study that is interested in both localization and classification is done by (Filkovic, 2014). In order to detect traffic signs Integral Channel Feature (ChnFtrs) detector is taken advantage of and for the recognition part CNN is used. A group of researchers created their own dataset which has 100000 images and applied detection and classification algorithm consisting of CNN structure (Zhu et al., 2016). As mentioned before, in ILSVRC14 competition the structure called GoogleNet which has 22 layer multi-scale inception network was created (Szegedy et al., 2014) and got the first place with 6.67 % error rate. Again in ILSVRC 2015, 152 layer CNN (He et al., 2015) took the 1^{st} place with 3.57 % error and this is the deepest structure hadn't been tried before. A study done in NY University (Sermanet et al., 2011) is accomplished using CNN with multi-scale convolutional network shown in Figure 2.1.

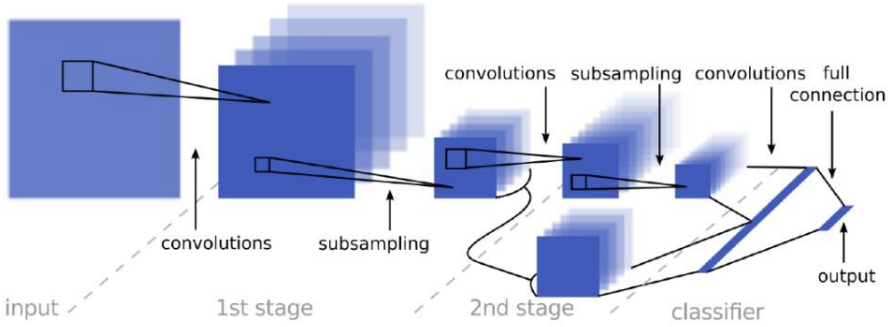


Figure 2.1 Network structure of Multi-Scale CNN (Sermanet et al., 2011)

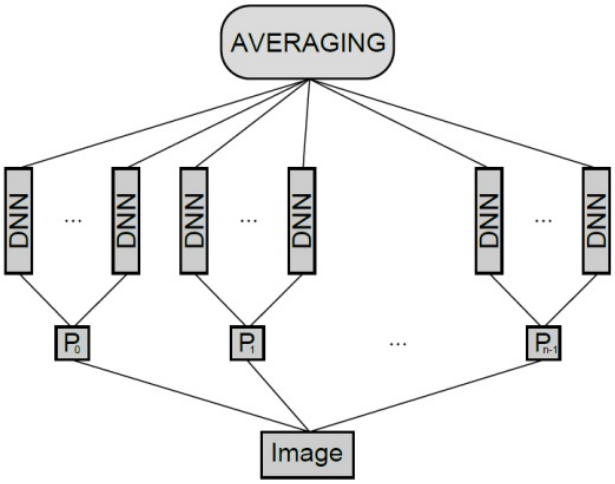


Figure 2.2 Network structure of averaging DNN (Ciregan et al., 2012)

The study by (Sermanet et al., 2011) uses classic CNN but the last layer of the network there is fully connected layer which has a different type of standard fully connected because also the outputs of other convolution layers added to the fully connected part of last convolution layer. This method can reach 99.17 % recognition rate and it is beyond the best human performance which is 98.81 %.

Another study that uses CNN is done by a group of institutes (Ciregan et al., 2012) shown in Figure 2.2 which uses lots of CNN structure at the same time and takes average of them getting 99.46 % accuracy rate.

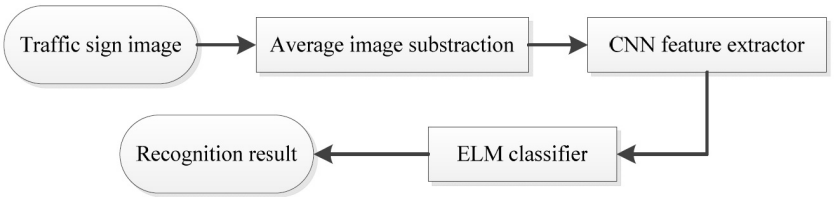


Figure 2.3 Network structure of CNN with ELM classifier (Zeng et al., 2015)

Comparing to the study done in NY University (Sermanet et al., 2011), researchers from National University of Defense Technology in China use a different configuration at the last layer of the network (Zeng et al., 2015) which is ELM classifier instead of fully connected layers and is shown in Figure 2.3. It is an inspiring work in the context of CNN and has 99.4 % recognition rate. A network similar to Sermanet is proposed

by (Wu, 2013), achieved 99.73 % performance in 'Danger' category and 97.62 % in 'Mandatory' category. The network structure is shown in Figure 5.

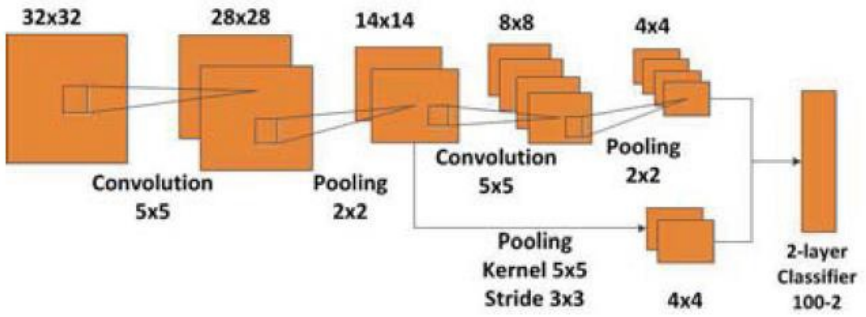


Figure 2.4 Network structure of CNN (Wu, 2013)

A recent study deals with sign recognition problem is also done with CNN which uses a method of residual network (Wen et al., 2017) and achieved performance of 99.66%. In this work, all layer may effect the last fully connected layer by addition as shown in Figure 2.5. For training, the dataset is augmented by rotating within $(-30°, 30°)$ and converting to YUV color format to minimize the various light conditions. In order to prevent overfitting, the number of training images is increased 21 times which results 741.048 images.

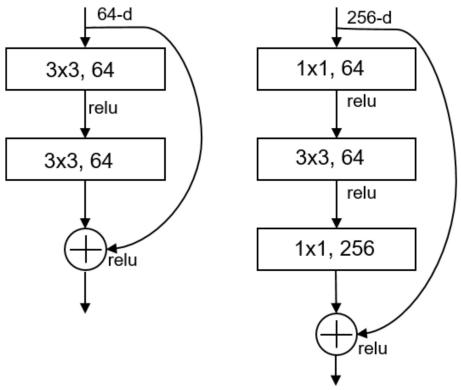


Figure 2.5 Modified residual network (Wen et al., 2017)

CNN was the main structure and traffic sign recognition was the main problem but in all studies mentioned before classes are given to the system directly so as to have 43 classes. In the study (Mao et al., 2016) the sign recognition problem was solved

by separating main classes into small families using CNN-oriented family clustering so they could be classified successfully much more than previous studies as achieving 99.67% accuracy rate. The hierarchical classification families are shown in Figure 2.6.

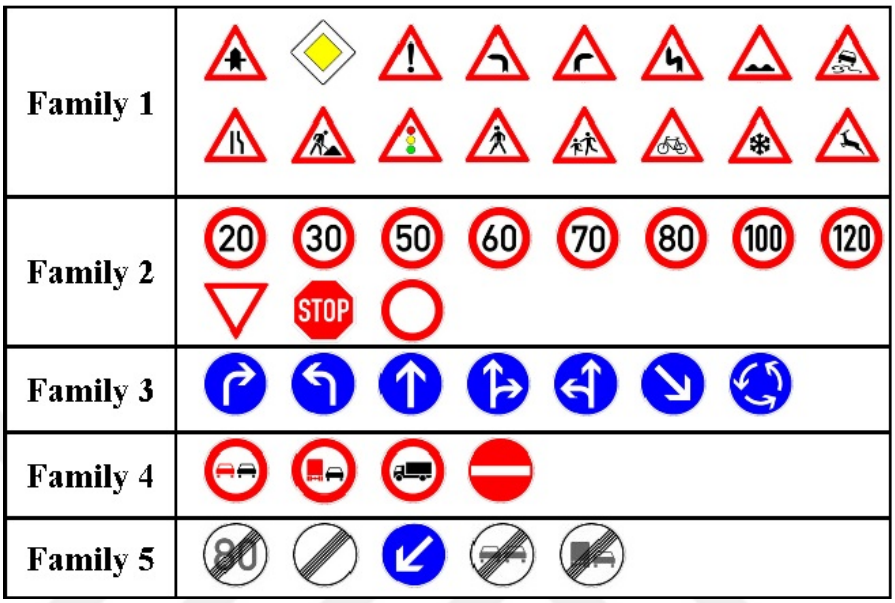


Figure 2.6 Hierarchical structure of CNN (Mao et al., 2016)

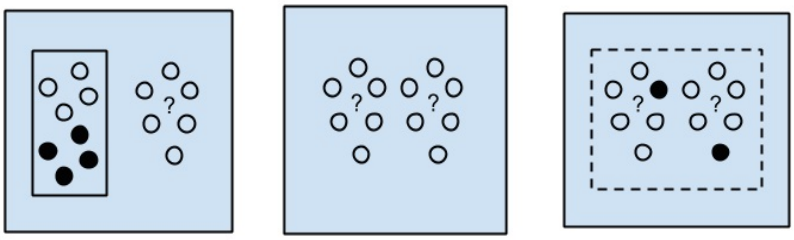
CHAPTER THREE
METHODOLOGY

3.1 Machine Learning

Before introducing CNN it is important to discuss some fundamentals of machine learning and its categories. Machine learning is a branch which aims to give the computer the ability to work as a decision-maker without explicitly programmed. In contrast to handcrafted methods which are simple machine vision techniques, machine learning mainly needs and takes advantage of huge amount of data in order to perform at the desired level. The most important point of machine learning systems is that it can learn the task from the training data. At the recent years, it has been widely used in many areas which need fast and accurate results such as object classifier, bank credit ratings, sentiment analysis in a sentence, detecting cancer cells, etc. Machine learning can be classified as learning style which will be told in next topics.

3.1.1 Machine Learning Algorithms by Learning Styles

Mainly machine learning can be divided into 3 categories in terms of learning style which are supervised, unsupervised and reinforcement learning as shown in Figure 3.1 and called according to the input-output relations of the system.



a) Supervised Learning | b) Unsupervised Learning | c) Reinforcement Learning

Figure 3.1 Machine learning categories (Brownlee, 2019c)

3.1.1.1 Supervised Learning

The structure in which inputs and labels of the training set are available is called supervised learning. Labels are the desired response of the system to a specific input. In this type of learning there is always a teacher to tell what should be the output of the system. By examining the target and the output of the system, it tells how much the system adapt itself to result in a correct direction so as to decrease the difference between the real output of the system and desired response. Classification and regression are examples of supervised learning algorithms.

3.1.1.2 Unsupervised Learning

Contrary to supervised learning there is no label for a system that aims to learn an input-output relation. Only the inputs are given and in this type of learning system can learn by itself by considering the similarities of the inputs according to their features. Clustering and dimensionality reduction are main examples of this category.

3.1.1.3 Reinforcement Learning

In this type of learning supervised and unsupervised learning are mixed since there are both labeled and unlabeled samples and it is expected that the system learns the patterns by making predictions and examining the results of it. Gaming algorithms are mostly taking advantage of this learning.

3.1.2 Most Known Machine Learning Algorithms

The algorithms that will be mentioned are categorized by their popularity. The criterion for grouping is the functions used in the system. There are couple of algorithms should be mentioned as listed below.

3.1.2.1 Artificial Neural Network Algorithms

ANN is one of the popular field for the community that studies regression and classification problems. The working principle is inspired by how human neural networks works as understood from the name. Main algorithms of ANN are listed as

- Perceptron
- Multilayer Perceptrons
- Hopfield Network
- Radial Basis Function Network (RBFN)
- Deep Learning Networks : Deep networks have structural similarity with ANN. Mainly they are constructed based on same logic but 'deep learning' term differs with ANN by 2 ways ; it is preferred in a system where features are learned by itself and layers are much more deeper compared with ML algorithms. Popular algorithms (Brownlee, 2019a) are
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Networks (RNNs)
  - Stacked Auto-Encoders
  - Deep Boltzmann Machine (DBM)
  - Deep Belief Networks (DBN)

3.1.2.2 Clustering Algorithms

Clustering is needed where a group of dataset has to be considered as separated classes. In this type of algorithm data is organized so as to have commonality. Mostly used algorithm are listed below.

- k-Means
- k-Medians
- Expectation Maximization (EM)
- Hierarchical Clustering

3.1.2.3 Regression Algorithms

Regression algorithms are appropriate for modeling a relation resulting a target value in terms of independent variable. Some of the algorithms are

- Linear Regression
- Logistic Regression
- Support Vector Machines
- Multiple Regression Algorithm

3.1.2.4 Bayesian Algorithms

Bayesian is a kind of algorithm in which any feature of the system have an equal weight and independent occurrence on the outcome of system which is based on probability. Most popular algorithms are

- Naive Bayes
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Bayesian Belief Network (BBN)
- Bayesian Network (BN)

3.2 Artificial Neural Network

As it is well known, Artificial Neural Network (mostly called Neural Network) is one of the common method used in machine learning. It has been inspired by biological system of human. The working principle is nearly the same as how brain works. In this thesis as supervised learning method, Convolutional Neural Networks (CNN) will be used. This algorithm is similar to Multi-Layer Perceptrons (MLP) but much more advanced. In methodology part, MLP and CNN structure will be clarified as the method of the thesis. ANN is a structure that models human brain in a way of great computation

ability and revealed in the middle of 20th century. It consists of considerable amount of neurons and connections between these neurons so as to compute both feedforward and backward propagation. Feedforward process is utilized to reach output value of the system and backward is to obtain effect of error term on each weight in order to update parameters of the model. There are some algorithms of ANN that will be mentioned called Perceptron and Multilayer Perceptrons in order to inform about base structure of the thesis.

3.2.1 Perceptron

Perceptron is the basic structure of neural network. It has multiple input and one output. The structure of perceptron is shown in Figure 3.2.

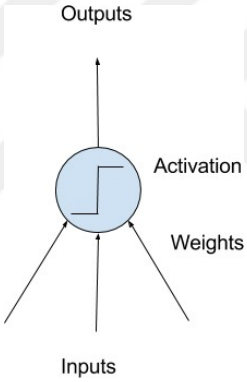


Figure 3.2 Perceptron (Brownlee, 2019b)

Inputs and weights are multiplied in matrix form and activation function is used to get output. An activation function is a simple mapping of summed weighted input to the output of the neuron. It is called an activation function because it governs the threshold at which the neuron is activated and strength of the output signal. If the network has more layer it would be Multi-Layer Perceptron which is shown in Figure 3.3.

3.2.2 Multilayer Perceptron

As seen in Figure 3.3, it may have one or more layer and the output can be multidimensional. First row is the input layer which has 3 neuron in this structure. Second row is hidden layer that has 4 neurons and last row is the output layer which

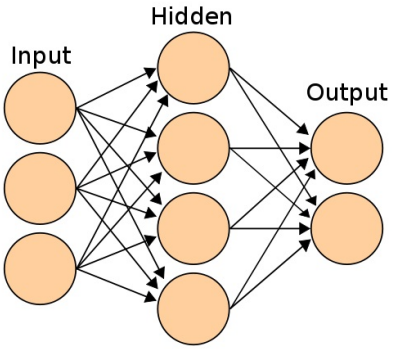


Figure 3.3 Multilayer perceptron (Pavlov, 2019)

has 3 neurons. Each neuron of the hidden layer is connected to both previous and next layers neurons and all have different weights. So each connection is called weight in this structure. Also each layer has an activation function in order to make system nonlinear to success on nonlinear classifier.

Having the output can be achieved by feedforward implementation. All the inputs are multiplied with weights and then added a bias value. Final value is processed through activation function and output of the layer is obtained. These process is repeated until the output of the network is obtained. At the beginning of the feedforward process all the weights are randomly initialized or determined by some unsupervised methods such as autoencoders to make the process much more faster. Because of that this is a supervised learning, there should be a label for each input. Cost function helps the weights updated for each epoch which means all the training data is given to the system just once. If the system are optimized 100 times it means weights are updated 100 times and all the data is processed 100 times.

Weight updating is a bit confusing part because calculating the effect of the cost function for each layer in the system is much more complex than it is thought and it needs huge amount of power for a big dataset. The way of updating these weights are done by backpropagation algorithm and it uses gradient descent method. Basically by starting from the output layer, the effect of the cost function is reflected backward to each weight of network as presented in Figure 3.4.

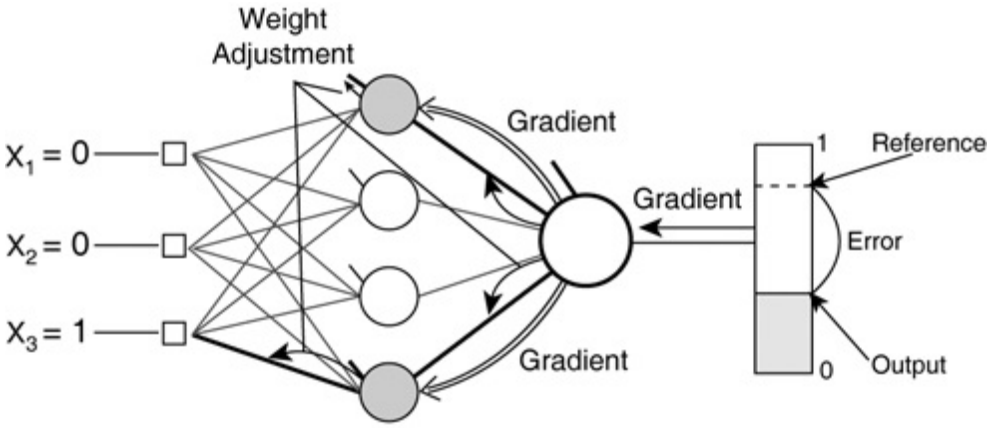


Figure 3.4 Backpropagation of MLP (Yaldex, 2019)

3.3 Convolutional Neural Network

CNN is a branch of deep neural network. It is a different version of MLP. The main goal is to train network and determine the filter weights. CNN is a feedforward structure. There are some mathematical concept of CNN which would be critic to learn :

- Convolution operation
- Subsampling operation (pooling)
- Classifier (fully connected layers)
- Backpropagation
- Batch Normalization
- Dropout

Convolutional Neural Network shown in Figure3.5 has similar base but feedforward process is done by convolving each input with described filter. Backpropagation approaching is the same as MLP to update weight of the filters in each layer.

3.3.1 Convolution Operation

Convolution in CNN means filtering in image processing. It is the first part of the network. Convolution is sliding the kernel (filter) over the entire two dimensional data one by one which is an operation commonly used in image processing. There are

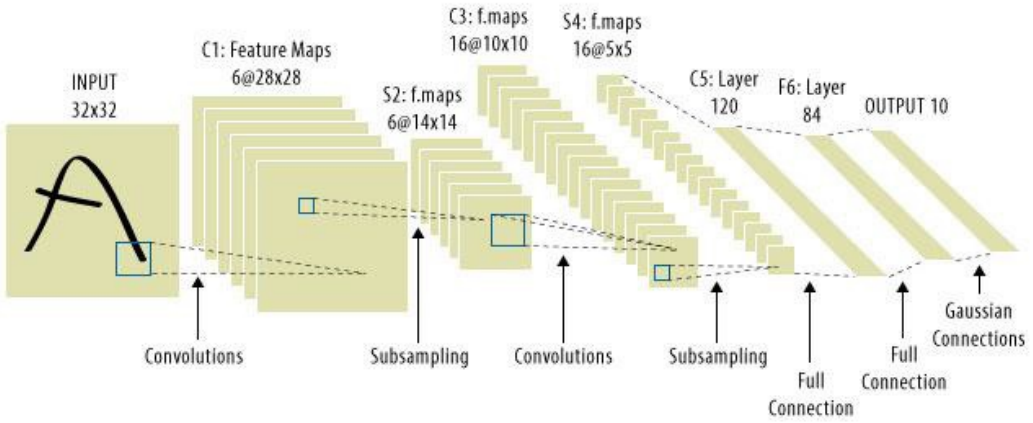


Figure 3.5 2-layers CNN structure (Wilson, 2019)

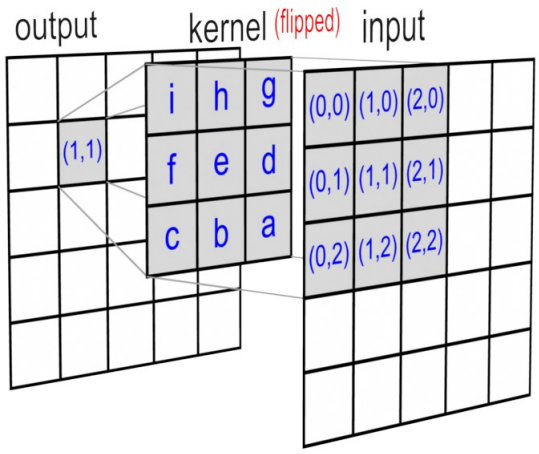


Figure 3.6 Convolution operation (Ahn, 2019)

critical parameters in this stage which are stride S, padding P, number of filters K and dimension of filter FxF. Weights can be randomly initialized or pretrained values.

- Stride S: how many pixels to be shifted
- Padding P: how many 0 valued pixels to be added to contour of two dimensional input
- Number of filters K
- Dimension of filter FxF

The example of how convolution works is shown in Figure 3.6 and Figure 3.7.

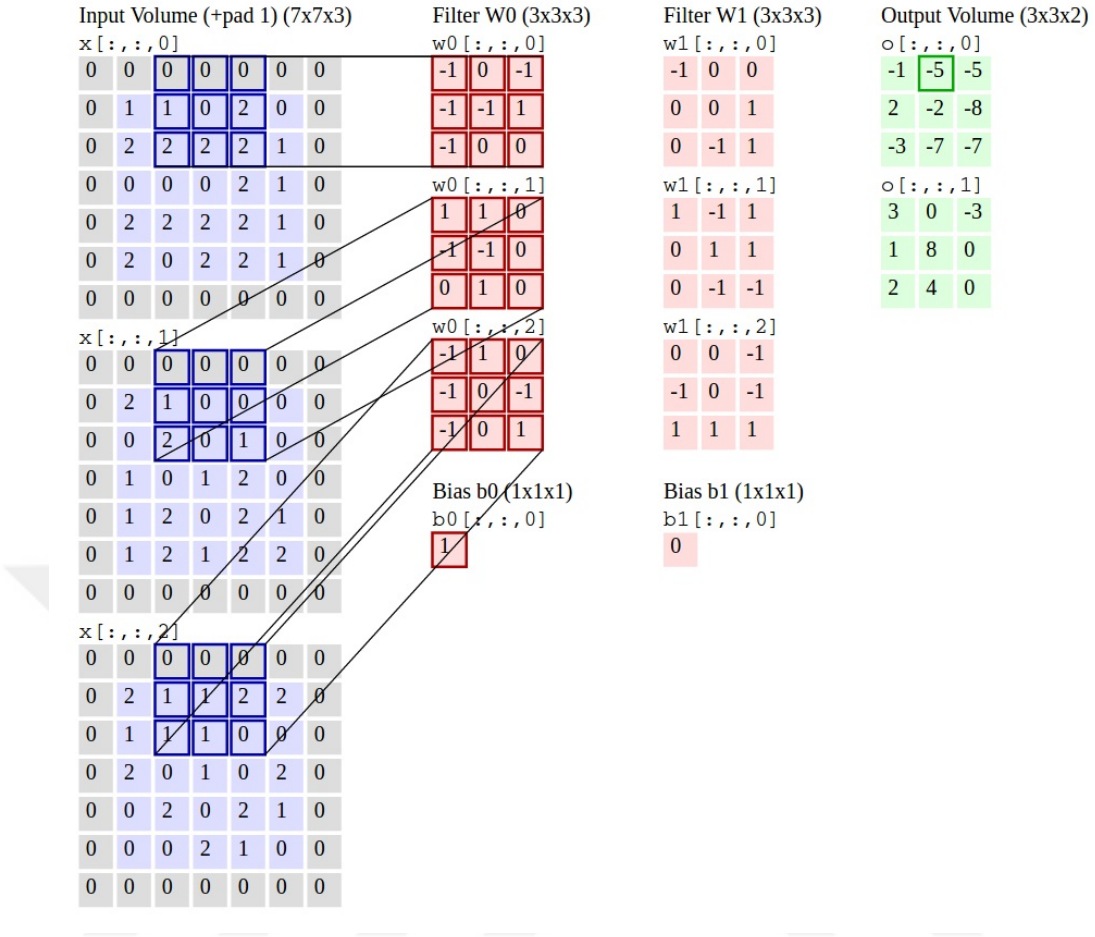


Figure 3.7 Convolution operation with 3 channels (dos Santos, 2019)

3.3.2 Pooling Operation

Pooling (subsampling) is used to reduce dimension of the output of convolutional layer shown in Figure 3.8 and make the system work much more faster and also helps shift invariance. There are two types of pooling namely max-pooling and average-pooling. Max-pooling takes the highest value of 2x2 box and average-pooling does it with the mean value of 2x2 box.

3.3.3 Fully Connected Layer

The convolutional layer is followed by fully connected layer which is a standard MLP structure and all the neurons between two consecutive layer are connected to each other so that is why it is called as fully connected layer as seen in Figure 3.9.

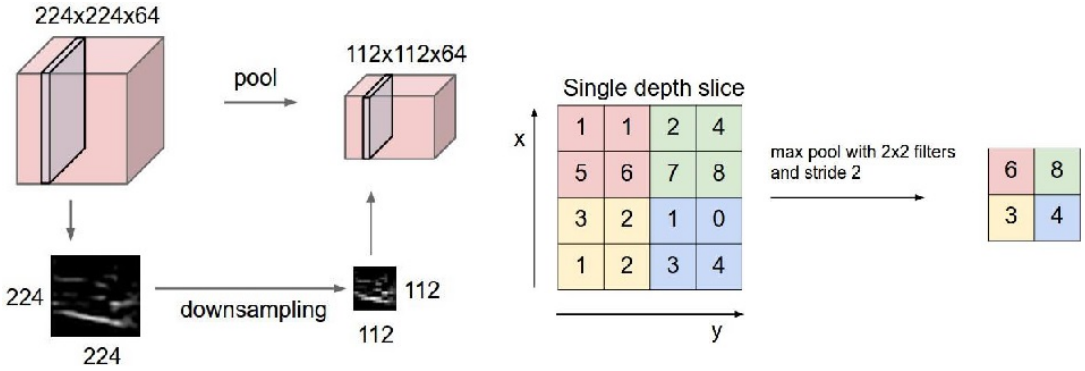


Figure 3.8 Pooling operation (Karpathy, 2019)

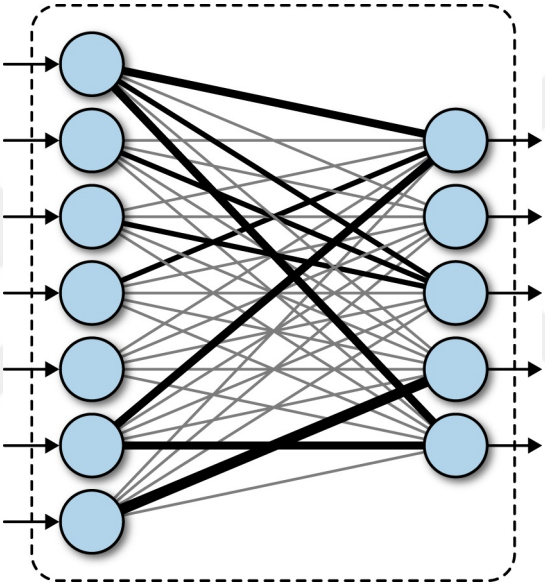


Figure 3.9 Fully connected layer (Zadeh, 2018)

3.3.4 Backpropagation in CNN

The crucial part of CNN is backpropagation as in all the deep learning methods. Weights of the system needs to be updated to make sure that the system learns input-output pattern. The way to realize that is backpropagating from output to the neurons of each layer of the network. The backpropagation formulas are given in Equation 3.3 and Equation 3.4.

Notation : The following notation will be used to clarify equations:

1. l is the lth layer where l=1 is the first layer and l=L is the last layer.
2. Input *o* has dimension HxW and has x by y as the iterators

3. Filter or kernel w is of dimension k_1xk_2 has x by y as the iterators
4. w^l is the weight matrix connecting neurons of layer l with neurons of layer l-1 .
5. b^l is the bias unit at layer l.
6. $o^l_{i,j}$ is the convolved input vector at layer l plus the bias represented as

$$o^l_{i,j} = \sum_x \sum_y w^l_{x,y} a^{l-1}_{i+x,j+y} + b^l \tag{3.1}$$

7. $a^l_{i,j}$ is the output vector at layer l given by

$$a^l_{i,j} = f(o^l_{i,j}) \tag{3.2}$$

8. $f(\cdot)$ is the activation function. Application of the activation layer to the convolved input vector at layer l is given by $f(o^l_{i,j})$

$$\begin{aligned} \delta^l_{x,y} = \frac{\partial E}{\partial o^l_{x,y}} &= \sum_{x'} \sum_{y'} \frac{\partial E}{\partial o^{l+1}_{x',y'}} \frac{\partial o^{l+1}_{x',y'}}{\partial o^l_{x,y}} \\ &= \sum_{x'} \sum_{y'} \delta^{l+1}_{x',y'} \frac{\partial o^{l+1}_{x',y'}}{\partial o^l_{x,y}} \end{aligned} \tag{3.3}$$

$$\begin{aligned} \frac{\partial o^{l+1}_{x',y'}}{\partial o^l_{x,y}} &= \frac{\partial}{\partial o^l_{x,y}} \Big(\sum_{x''} \sum_{y''} w^{l+1}_{x'',y''} a^l_{x'-x'',y'-y''} + b^{l+1} \Big) \\ &= \frac{\partial}{\partial o^l_{x,y}} \Big(\sum_{x''} \sum_{y''} w^{l+1}_{x'',y''} f(o^l_{x'-x'',y'-y''}) + b^{l+1} \Big) \\ &= \frac{\partial}{\partial o^l_{x,y}} \Big(w^{l+1}_{0,0} f(o^l_{x'-0,y'-0}) + \ldots + w^{l+1}_{x'-x,y'-y} f(o^l_{x,y}) + \ldots + b^{l+1} \Big) \\ &= \frac{\partial}{\partial o^l_{x,y}} \Big(w^{l+1}_{x'-x,y'-y} f(o^l_{x,y}) \Big) \\ &= w^{l+1}_{x'-x,y'-y} \frac{\partial}{\partial o^l_{x,y}} (f(o^l_{x,y})) \\ &= w^{l+1}_{x'-x,y'-y} f'(o^l_{x,y}) \end{aligned} \tag{3.4}$$

Arranging Equation 3.3 and Equation 3.4 results in Equation 3.5 which will be utilized in next steps.

$$\begin{aligned}\frac{\partial E}{\partial o^{l}_{x,y}} &= \sum_{x'}\sum_{y'} \delta^{l+1}_{x',y'} w^{l+1}_{x'-x,y'-y} f'(o^{l}_{x,y}) \\ &= \delta^{l+1}_{x',y'} * w^{l+1}_{-x,-y} f'(o^{l}_{x,y}) \\ &= \delta^{l+1}_{x',y'} * rot_{180}(w^{l+1}_{x,y}) f'(o^{l}_{x,y})\end{aligned} \tag{3.5}$$

An error term is defined which is derivative of cost function E to the output of convolution operation at layer l and last term which is error term of layer l is written in terms of error term in layer $l+1$. There it can be seen that both for forward and backward propagation convolution operation is used but there is difference between forward and backward which is the type of convolution and shown in Figure 3.10. While propagating in forward direction valid or same convolution are used but for propagating in backward direction full convolution is used. The forward and backward convolutions are explained in Figure 3.11.

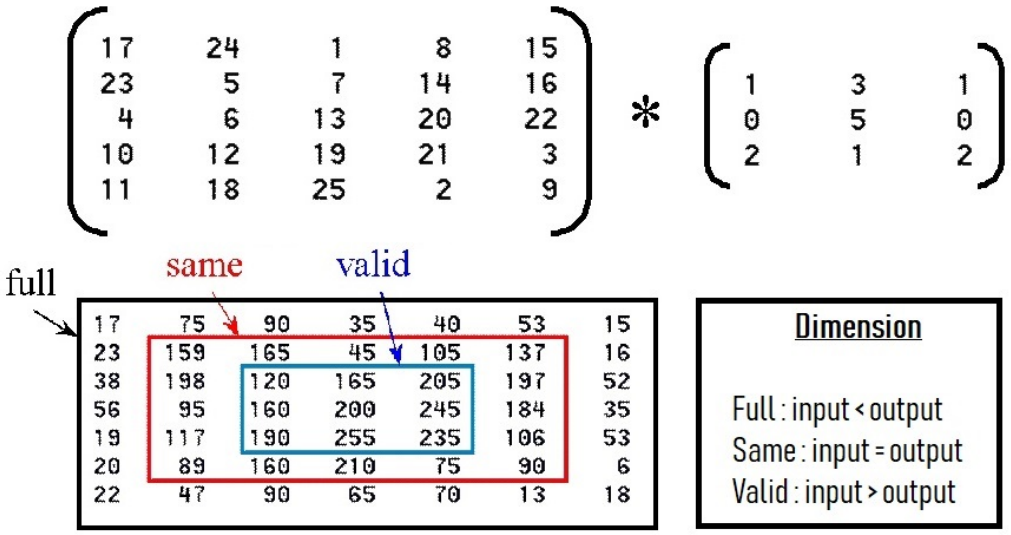


Figure 3.10 Types of convolution (Loomis, 2019)

Weight updating should be considered after extracting error term. From the Equation 3.3 it has been figured out that the error term at layer L can be determined by the error term at layer L+1. This will help update weights when backpropagating from the output of the network. Rest of the equations are given as Equation 3.6, 3.7 and 3.8. Equation

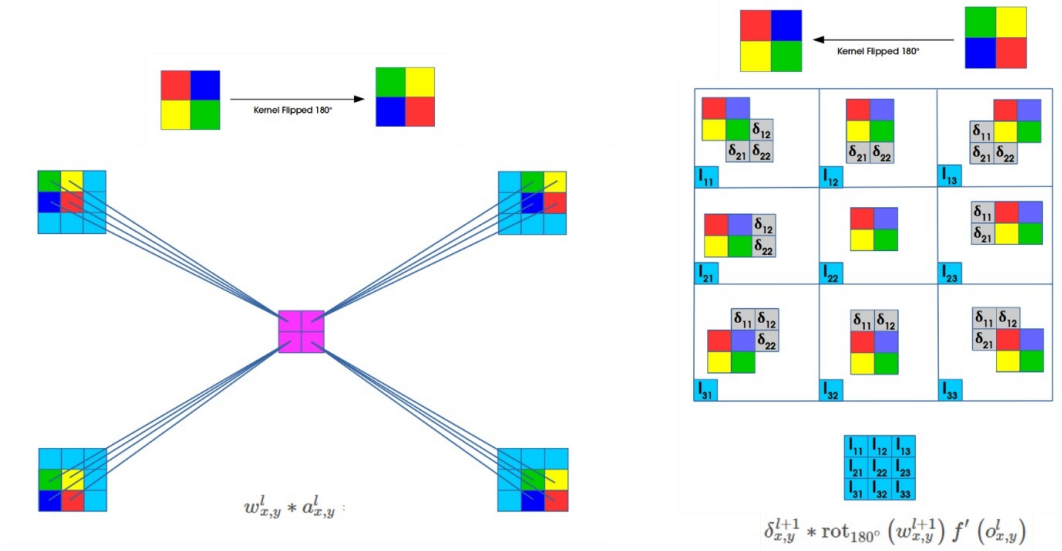


Figure 3.11 Forward and backward propagation for convolution operation (Kafunah, 2019)

3.9 is the final expression for update rule.

$$\begin{aligned}\frac{\partial E}{\partial w^l_{x,y}} &= \sum_{x'}\sum_{y'} \frac{\partial E}{\partial o^l_{x',y'}} \frac{\partial o^l_{x',y'}}{\partial w^l_{x,y}} \\ &= \sum_{x'}\sum_{y'} \delta^l_{x',y'} \frac{\partial o^l_{x',y'}}{\partial w^l_{x,y}}\end{aligned} \tag{3.6}$$

$$\begin{aligned}\frac{\partial o^l_{x',y'}}{\partial w^l_{x,y}} &= \frac{\partial}{\partial w^l_{x,y}}\left(\sum_{x''}\sum_{y''} w^l_{x'',y''} a^l_{x'-x'',y'-y''} + b^l\right) \\ &= \frac{\partial}{\partial w^l_{x,y}}\left(\sum_{x''}\sum_{y''} w^l_{x'',y''} f(o^{l-1}_{x'-x'',y'-y''}) + b^l\right) \\ &= \frac{\partial}{\partial w^l_{x,y}}\left(w^l_{0,0} f(o^{l-1}_{x'-0,y'-0}) + \ldots + w^l_{x,y} f(o^{l-1}_{x'-x,y'-y}) + \ldots + b^l\right) \\ &= \frac{\partial}{\partial w^l_{x,y}}\left(w^l_{x,y} f(o^{l-1}_{x'-x,y'-y})\right) \\ &= f(o^{l-1}_{x'-x,y'-y})\end{aligned} \tag{3.7}$$

$$\begin{aligned}\frac{\partial E}{\partial w^l_{x,y}} &= \sum_{x'}\sum_{y'} \delta^l_{x',y'} f(o^{l-1}_{x'-x,y'-y}) \\ &= \delta^l_{x,y} * f(o^{l-1}_{-x,-y}) \\ &= \delta^l_{x,y} * f(rot_{180}(o^{l-1}_{x,y}))\end{aligned} \tag{3.8}$$

$$\begin{aligned}w^l_{x,y}(n+1) &= w^l_{x,y}(n) - \eta\left(\frac{\partial E}{\partial w^l_{x,y}(n)}\right) \\ &= w^l_{x,y}(n) - \eta\left(\delta^l_{x,y} * f(rot_{180}(o^{l-1}_{x,y}))\right)\end{aligned} \tag{3.9}$$

n is iteration number and η is learning rate. So it can be seen that forward and backward operations deal with the convolution operation. Either for forward and backward propagation, standard filtering and matrix multiplication operations are used but filters are rotated 180° when updating weights of the network as shown in Equation 3.9.

After all iterations are done and weights are not updated anymore, the network is ready to evaluate the test set. So Figure 3.12 shows an example of how network gives the information about the classes learned in training phase.

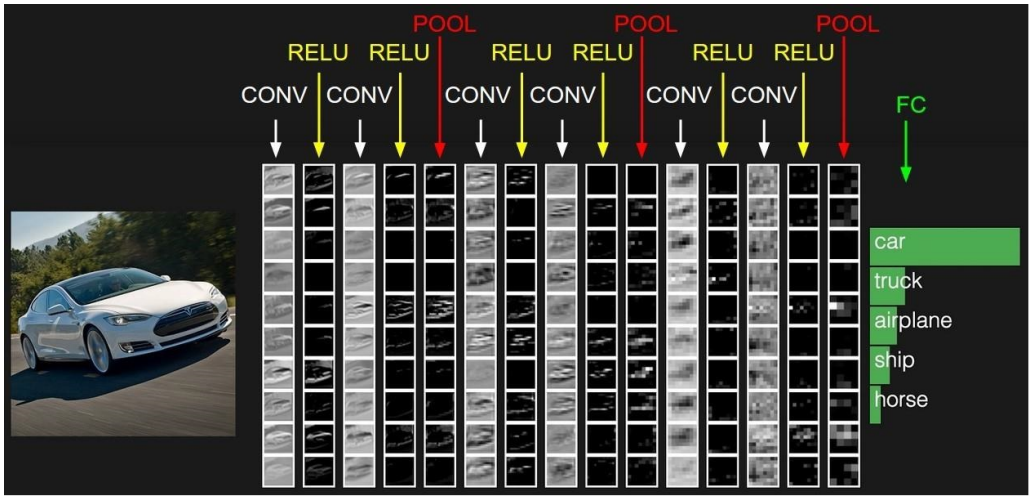


Figure 3.12 Test sample for CNN (Karpathy, 2019)

3.3.5 Batch Normalization

Batch normalization is a critical operation when big data is held and training time is expected to be long. Generally it is used to normalize the parameters after each convolution operation is done. It takes the mini-batch of convolution layer output and normalize them to prevent dealing with higher size of neurons. Equation 3.10 and 3.11 represents mean and variance and normalization are shown in Equation 3.12. After normalization, scale and shift operations are applied as given in Equation 3.13 (Ioffe et al., 2015).

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.10}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{3.11}$$

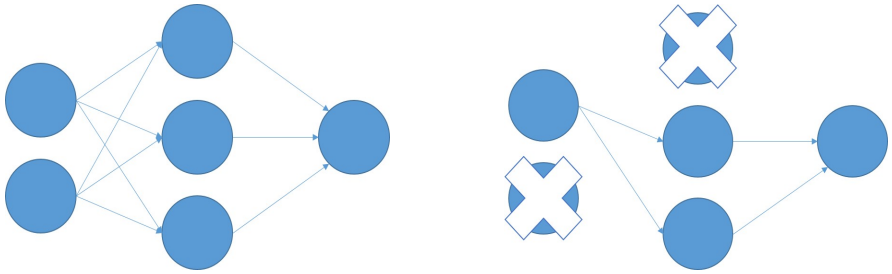
$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{3.12}$$

$$y_i = \gamma \hat{x_i} + \beta \tag{3.13}$$

where x is the output of convolution, m is the number of neurons belongs to layer and γ, β are coefficients for normalization.

3.3.6 Dropout

Dropout is a useful technique for CNN as an operation decreasing computing time and overfitting probability. It works as some output are dropped at any layer and all the connections with this output are biased so as to be zero. Dropout may get a value between 0 and 1 (0 is excluded) and indicating probability of cutting out outputs. Dropout operation is shown in Figure 3.13.



(a) Without dropout (b) With dropout

Figure 3.13 Dropout technique

3.3.7 Activation Functions

As applied in all machine learning methods, activation functions are essential points of both feedforward and backward process of learning. They determine how the output of the convolution layer are shaped. As mentioned in (Nwankpa et al., 2018) the advantages and disadvantages will be examined by comparing them with different networks.

3.3.7.1 Rectified Linear Units (ReLU)

ReLU is the most used function in ML studies because of fast convergence and easiness of getting derivative to backpropagate the system. ReLU function and its derivative are shown in Figure 3.14.

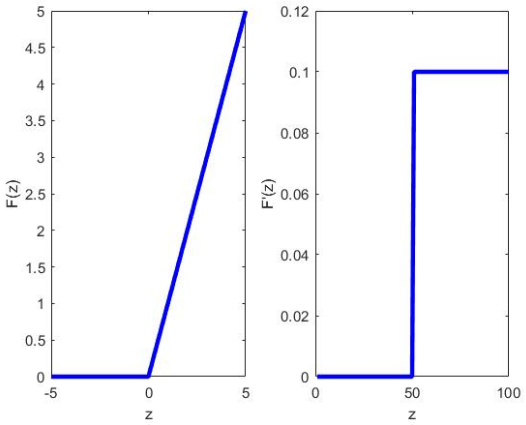


Figure 3.14 ReLU function and its derivative

$$F(z) = max(0, z) \tag{3.14}$$

$$F^{'}(z) = 1 \tag{3.15}$$

3.3.7.2 Sigmoid Function

The reason why sigmoid function is chosen is it is between 0 and 1 so it is appropriate for the case where the probability of anything shouldn't be under 0 and above 1. Only thing about sigmoid function is that system get stuck in training due to approximating strong negative values to zero. Sigmoid function and its derivative are shown in Figure 3.15.

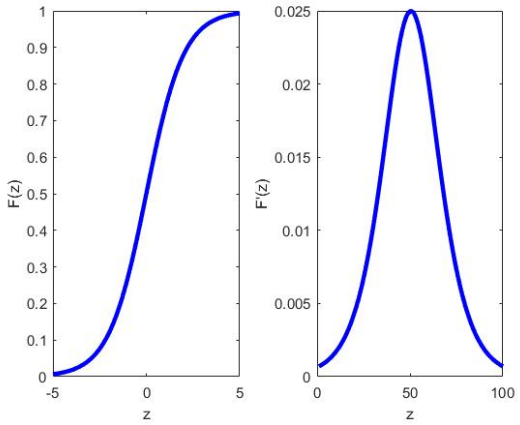


Figure 3.15 Sigmoid function and its derivative

$$F(z) = \frac{1}{1+e^{-z}} \tag{3.16}$$

$$F^{'}(z) = F(z)(1-F(z)) \tag{3.17}$$

3.3.7.3 Hyperbolic Tangent Function

Difference between sigmoid and hyperbolic tangent is that hyperbolic tangent gives an output between (-1,1). This function may be used to get over with stuck problem in sigmoid function. Hyperbolic tangent function and its derivative are shown in Figure 3.16.

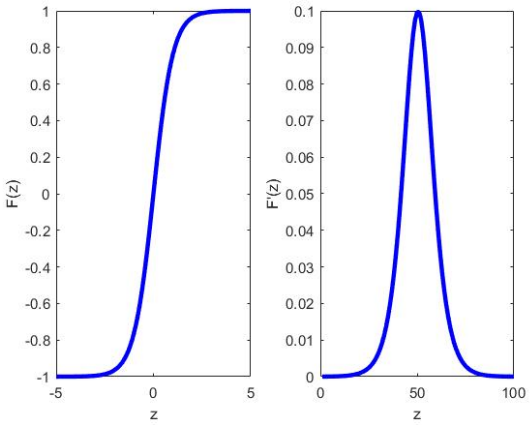


Figure 3.16 Hyperbolic tangent function and its derivative

$$F(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.18}$$

$$F^{'}(z) = (1 - F^{2}(z)) \tag{3.19}$$

3.3.7.4 Softmax Function

Softmax is a probability function so as to have a value 1 in sum no matter how inputs are big or negative or positive. It is generally used in the last fully connected layer when a classification is needed.

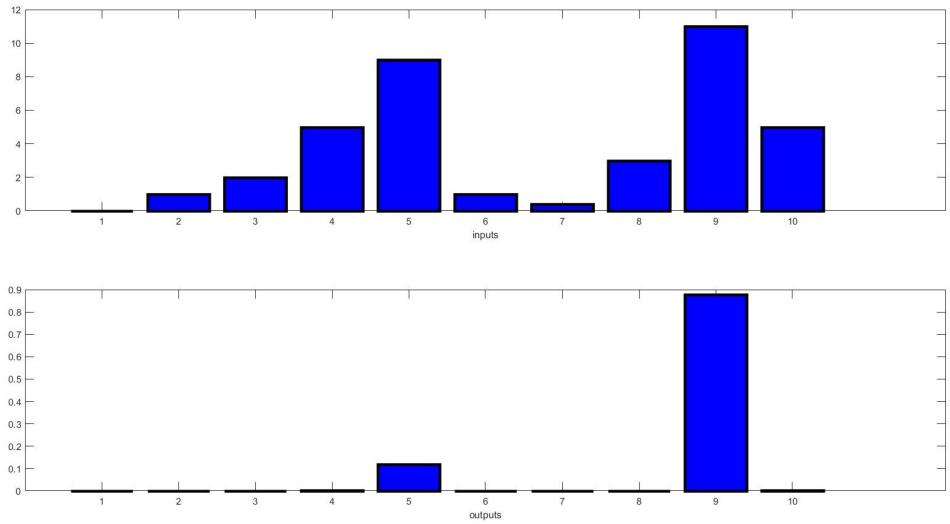


Figure 3.17 Softmax function

$$S(a): \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} \rightarrow \begin{pmatrix} S_1 \\ \vdots \\ S_N \end{pmatrix}$$

where S is the function, a_N is N^{th} element of input vector, S_N is N^{th} element of output vector.

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \tag{3.20}$$

3.4 Capsule Network

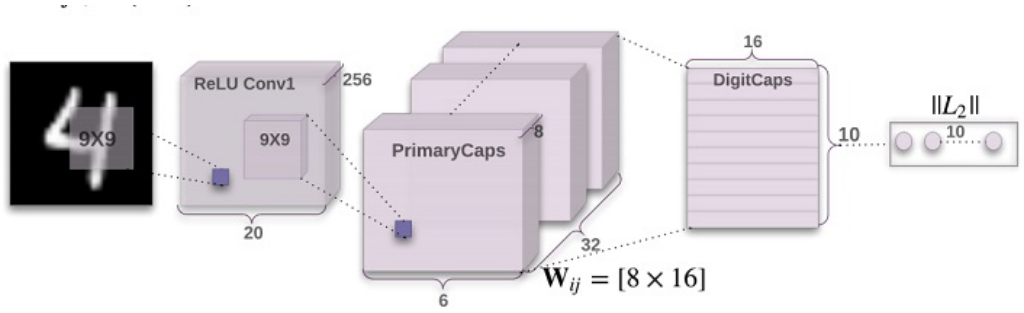


Figure 3.18 Capsule network (Rimal, 2019)

Capsule Network of which general structure is illustrated in Figure 3.18 is an advance form of machine learning which has common points with CNN but at some points there are differences about how propagating is done. The generalization capability of CNN is beyond almost all the machine learning algorithms but in some cases a model is needed which is much more invariant to the location of the inputs. Fundamentally CNN are excited with a group of neurons while capsule networks does it with a group of vector. This vector may contain a multiple feature of the input besides CNN has only a scalar value of a single neuron. Thus features like probability, pose and size are stored in this vector and combined with another group of vector which also has similar features for different patterns. A clarifying visual of comparison is shown in Figure 3.19.

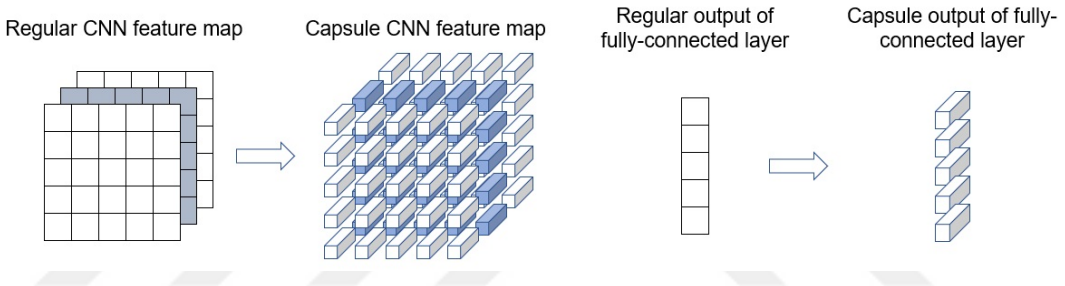


Figure 3.19 Capsule network vs CNN (Zhou et al., 2019)

CHAPTER FOUR
PROPOSED METHOD

4.1 GTSRB Dataset

Figure 4.1 GTSRB classes samples (Hijazi et al., 2019)

There are many studies in the literature that deals with creating a system for detection and recognition of traffic signs. To compare these studies with each other it is essential that they take advantage of same dataset. There are couple of datasets available as benchmark and most of them are from corporations working in Europe. GTSRB (German Traffic Sign Recognition Benchmark) is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) (Stallkamp et al., 2012). Because of that the variety of images are fair enough to train a network and having 39209 training and 12630 test images from 43 class makes this dataset a benchmark for studies looking for a dataset for traffic sign classification. Dataset consisting images from 15x15 to 250x250 are taken as 'ppm' format originally. These images are cropped from a video which was taken in Germany and representative images of each class are shown in Figure 4.1. Train and test labels are given in CSV format including ROI which is the exact location of sign in image. Figure 4.2 shows how label annotations are done using specific program.

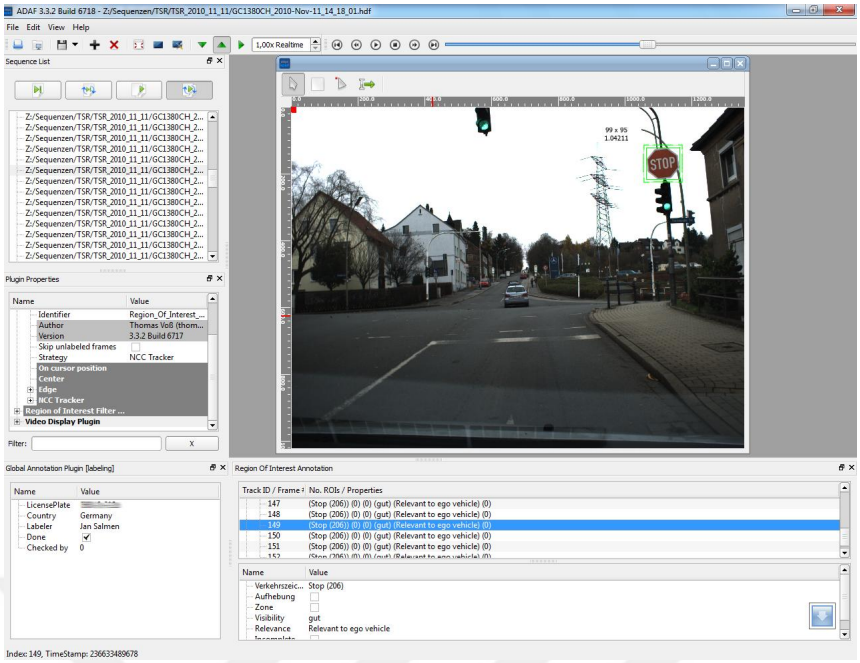


Figure 4.2 GTSRB annotations (GTSRB, 2019)

Number of training images of dataset would be fair enough to train the system if all classes have a balanced number of images but some classes may have 210 images while the other one has 2100. So in order to get a homogeneous dataset some augmentation techniques has been applied which are rotation, skew, scale and crop operations. After all operations, 5000 number of images are obtained for each class for training and that means 215000 images in total. To train the system adequately all the images are resized to 32x32.

4.2 Hierarchical Classification

Figure 4.3 GTSRB classes (Ellahyani et al., 2016)

Most of the classifiers proposed for the traffic sign recognition deal with flat classification where each sign image is assigned to one of the 43 classes. But from Figure 4.3, it can be seen that the images can be collected into subcategories. So the hierarchical classification can be more appropriate. In hierarchical classification dataset is divided into groups so that the classes in the same group are more similar to each other. The main problem in hierarchical classification is the decision of the number of subgroups and their separation process. For the solution of this problem hierarchical clustering can be used. Hierarchical clustering groups the data according to a similarity measurement and creates a cluster tree. Pair of classes are linked according to their similarity extracted with the linkage algorithm.

Linkage algorithm is used for clustering (Sayad, 2019) by using similarity matrix of classes to be clustered hierarchically.
The clustering procedure;

- Find the shortest distance between elements (x_a, x_b)
- Group them as a new class S and assign their new distance value as $\max\left(d(S,x_1), d(S,x_2)..., d(S,x_n)\right)$
- Go on until all elements are classified

where $\rightarrow (x_1, x_2, ..., x_n)$ are SSIM values of representative images, $d((x_a, x_b))$ are distances, S is new class.

A small example of how linkage clustering works is represented in Figure 4.4. A dendrogram is a diagram that shows the hierarchical relationship between groups to be clustered. Figure 4.5 shows hierarchical similarity between features.

	a	b	c	d	e
a	0	17	21	31	23
b	17	0	30	34	21
c	21	30	0	28	39
d	31	34	28	0	43
e	23	21	39	43	0

	(a,b)	c	d	e
(a,b)	0	30	24	23
c	30	0	28	39
d	24	28	0	43
e	23	39	43	0

	((a,b),e)	c	d
((a,b),e)	0	39	43
c	39	0	28
d	43	28	0

	((a,b),e)	(c,d)
((a,b),e)	0	43
(c,d)	43	0

Figure 4.4 Linkage clustering

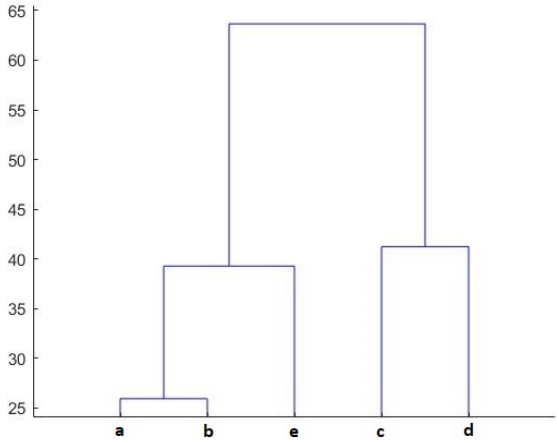


Figure 4.5 Dendrogram of clustering

4.3 Structure Similarity Index Measurement (SSIM)

For each classes an image that represents the whole class is needed for the calculation of the similarity between any two classes. Firstly these images were thought to be the average image of the class in dataset but taking their average wasn't enough because each class has different number of training images and each of them aren't captured with the same lighting condition. So this was not a satisfying solution and it has been decided that the best way to represent a class is using formal drawing which is also shown in traffic signs and Figure 4.7 shows a representative image of each class. Representative images will be used to determine the similarity between classes.

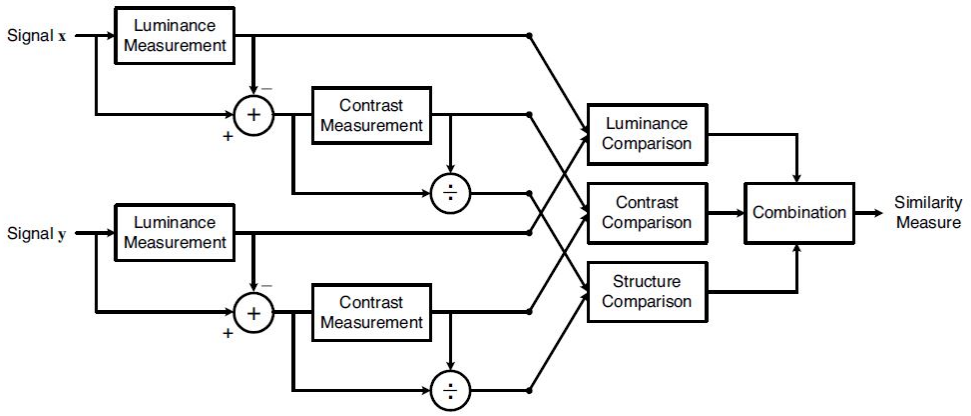


Figure 4.6 SSIM diagram (Wang et al., 2004)

In the structural similarity method in which the image representatives of the two classes are given as input, the output value, which is the product of brightness, contrast, and structural indices and expressed as in Equation 4.1 (Wang et al., 2004). Here μ_x and μ_y represent the image averages, σ_x and σ_y are standard deviations, and σ_{xy} is the cross-covariance values. C_1 and C_2 are constants added in case the multiplication is zero or infinity.

Figure 4.7 GTSRB representative classes

$$ssim(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.1}$$

To be able to create SSIM matrix, representative images of each classes calculated in MATLAB (MATLAB, 2016) and applied similarity index measurement.

4.4 Proposed Method

It has been mentioned that not a standard classification will be held in this thesis. In standard classification there are 43 classes and the system is taught to learn how to discriminate these classes. As a different method, hierarchical structure is proposed and main look can be seen in Figure 4.9. So the questions about how many subclasses should be and how it can be determined are all related with SSIM and linkage-clustering operations. All the similarity measurements of representative images shown in Figure 4.7 are calculated and given to the linkage-clustering function. As a result, hierarchical structure has been obtained as presented in Figure 4.8. Cutoff value is determined so as to be 2 and 4 main groups have been created for the first grouping stage.

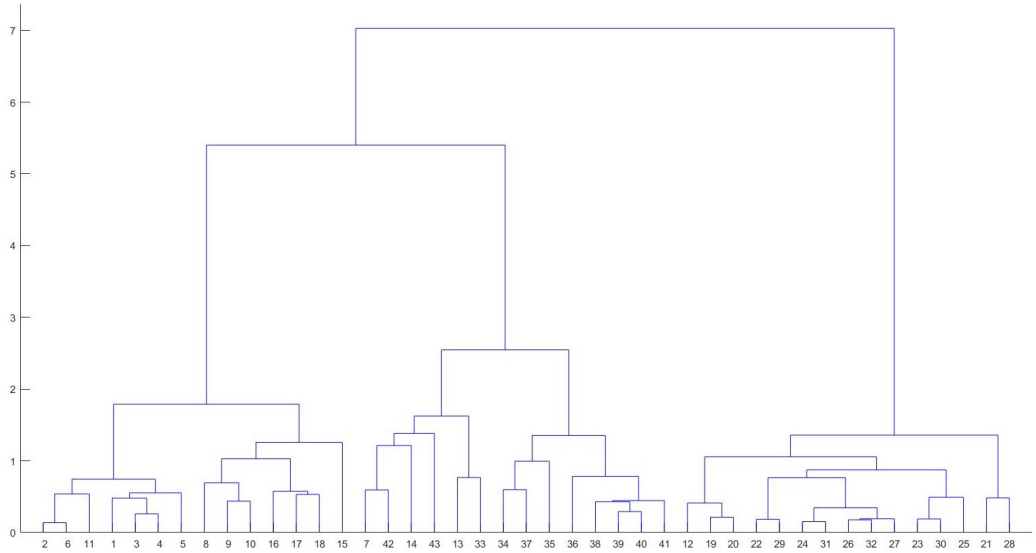


Figure 4.8 Dendrogram of linkage clustering

As it can be seen from Figure 4.9, the proposed hierarchical structure consists of five classifiers in total. Because of their proven success, CNNs are chosen as classifiers. CNN at first level divides the classes into four main subgroups and the CNNs at the second level classify the subgroups created at first part.

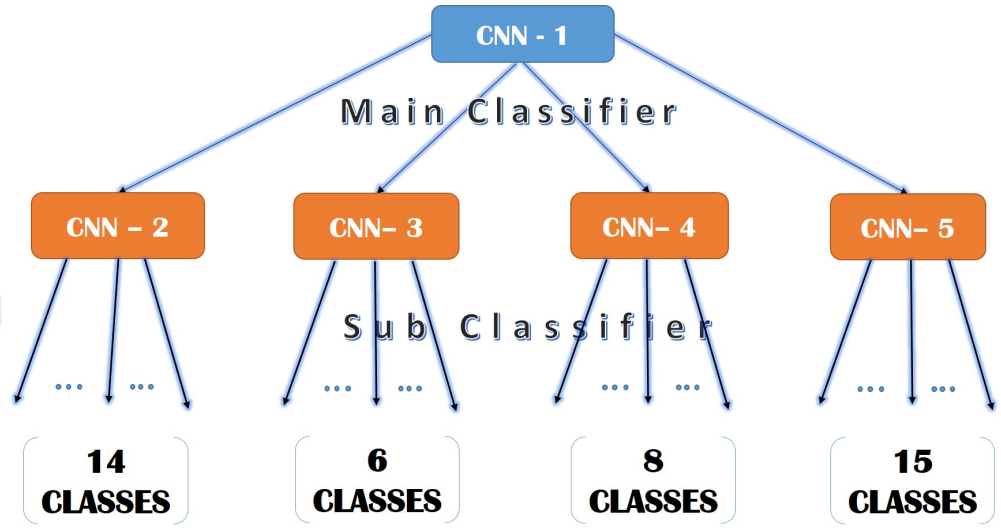


Figure 4.9 Hierarchical main classifier and subclassifiers

In order to reach the optimal CNN structure of the study, lots of networks are evaluated. This evaluation process mostly deals with how parameters affect the system for both accuracy and training time. Classical CNN algorithm are taken advantage of in this study. It has 4 convolutional layer and 1 fully connected layer at the end. Batch normalization and dropout techniques are the comparison issues but only dropout is used in the proposed structure. Parameters are directly related with training time in direct proportion so in terms of GPU capacity it should be adjusted well. Total parameter number of the model is 1873.000 and it can be seen in Figure 4.10. Activation functions are preferred as ReLU and dropout is applied after each convolution layer by the rate of 0.5.

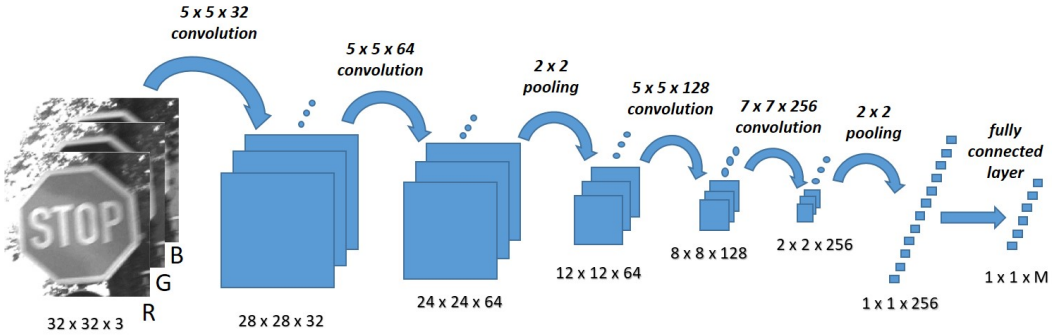


Figure 4.10 CNN structure of proposed method

CHAPTER FIVE
APPLICATION AND RESULTS

Before the extraordinary developments in graphic processors in recent years, such studies were carried out with standard image processing methods. Firstly, it has been studied how to make the model preprocessed with a standard image processing study before machine learning methods. Actually this was intended to be the pre-hierarchical structure by looking at the main properties of image like color and shape. For this purpose, different image processing techniques have been applied on MATLAB in terms of color and shape, but the fact that the data set contains very different lighting conditions and having low resolution on considerable number of images prevented from establishing a healthy structure. So after failure of this step it has been decided that pre-classification with computer vision techniques are time consuming and not appropriate for the study. As a conclusion CNN as a deep learning method is thought to be used in the study.

In order to see the effect of hierarchical and flat classification on performance in the recognition problem of traffic signs, a CNN structure was designed which separates the given data into 43 classes. To observe the effect of hierarchical classification on performance, a structure was created that hierarchically classifies traffic signs according to their colors. In this section, the performance of the proposed structure is compared with the performance of the proposed methods in the literature and comparison results are given. All the comparisons are realized with the structure given in Figure 4.10. To clarify how CNN processes within forward propagation layers are visualized after each convolution operation step by step.

5.1 Performance Comparison

5.1.1 Flat Classification

In order to compare the study, firstly, a structure was created which has only one network and that network classifies main 43 classes as default. No hierarchical structure is included so no subgroups are created. This allows to see how well the network is created, regardless of the hierarchical structure, succeeds in flat classification. The result was fair enough for study network and yielded 97.1% accuracy.

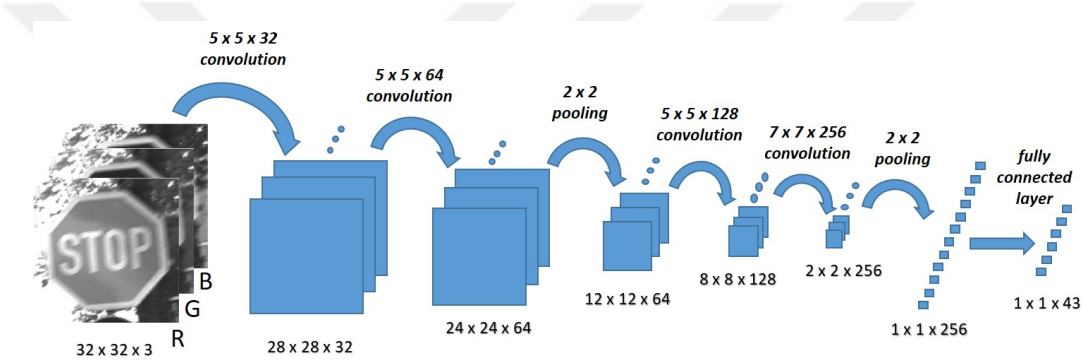


Figure 5.1 Flat CNN structure with 43 class output

5.1.2 Hierarchical Classification Based on Colors

After the image processing did not yield a healthy result, it was decided to construct this hierarchical structure according to the colors which are the most prominent features of the signs. The groups were determined as in Table 5.1 and 97% accuracy rate was reached in the study with these classes. However, this structure is not sufficient for the finalization of the thesis because determination of how these color based subgroups are formed is just a decision taken by visual perspective.

Table 5.1 Hierarchical classification based on colors

FAMILY 1	
FAMILY 2	
FAMILY 3	
FAMILY 4	

5.1.3 Hierarchical Classification by Grouping Algorithm

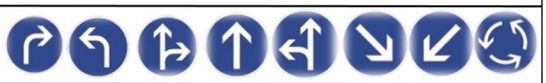
It has been mentioned in introduction that there will be two different hierarchical classification examined in thesis. The first one is achieved by taking advantage of linkage clustering method and the second one is done by using grouping algorithm (Surucu et al., 2018) shown in Algorithm 1 that is similar to linkage clustering. Classes as an output of this classification are shown in Table 5.2. Average images which is used in algorithm is given in Figure 5.2. The performance of this system is 97.8%.

Figure 5.2 Average images of GTSRB classes (Surucu et al., 2018)

Algorithm 1 : grouping algorithm

$MI(m) \equiv$ average image of classes, $m = 1:43$
for $i \leftarrow 1:43$ **do**
  for $j \leftarrow 1:43$ **do**
    $\phi_{ij} = ssim(MI(i), MI(j))$
  end for
end for
find A and B classes which has the highest dissimilarity rate and assign them as two different group $\rightarrow \phi_{AB} = \min_{i,j} \phi_{ij}$
$G_1 = \{A\}$, $G_2 = \{B\}$
s = 2 $\rightarrow$ group number
Calculate $V_m(t)$ belongs to each group for each $MI(m)$ image, t=1,..,s
for $m \leftarrow 1:43$ **do** (m $\neq$ A,B)
  for $t \leftarrow 1:s$ **do**
    c_t : number of class of group t,
    k : element index of group t
    $V_m(t) = \frac{1}{c_t} \sum_{k=1}^{c_t} \phi_{mk}$
  end for
  Find maximum $V_m\{t\}$ value and its group
  V_max = $max\{V_m(r)\}$, r : 1,..,s
  r = $argmax\{V_m(r)\}$
  if V_max $> \varepsilon$ and *and* std$\{\phi_{m,G_r\{k\}}\} < \beta$, $k = 1,..,c_r$ **then**
    group r $\leftarrow$ class m
  else
    s = s+1
    group s $\leftarrow$ class m
  end if
end for

Table 5.2 Hierarchical classification by grouping algorithm (Surucu et al., 2018)

FAMILY 1	
FAMILY 2	
FAMILY 3	
FAMILY 4	
FAMILY 5	
FAMILY 6	
FAMILY 7	
FAMILY 8	
FAMILY 9	

5.1.4 Hierarchical Classification Based on Confusion Matrix

Previously, it has been mentioned that the hierarchical classification of the proposed method is done by using SSIM and the input data for SSIM is the representative images of 43 classes in GTSRB dataset. Hierarchical classification may also be realized by examining confusion matrix of flat classifier. Extracting confusion matrix helps figuring out which classes are similar or not. By getting differences of each row, the new similarity matrix can be created and taking advantage of linkage clustering, dendrogram of the classes are observed so the new network can be trained according to the hierarchical classification. Table 5.4 shows the confusion matrix for 43-class flat classification. It is assumed that each class is represented as row vector in confusion matrix. By normalizing and taking the L2 distance between each classes(Godbole, 2002), similarity matrix can be obtained. Dendrogram of linkage-clustering is presented in Figure 5.3.

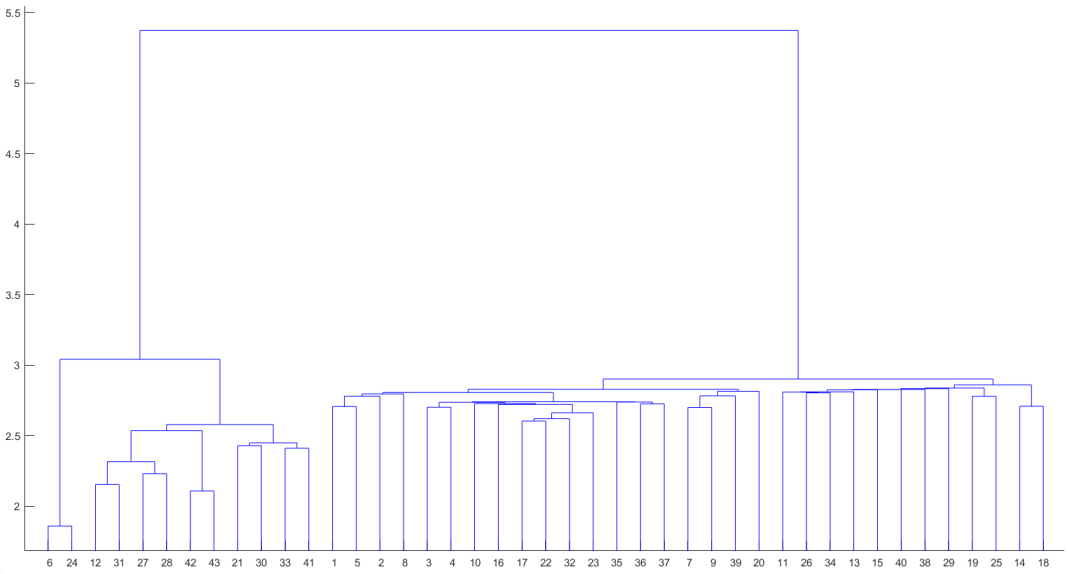


Figure 5.3 Dendrogram of clustering

The subclasses are created according to dendrogram so there are 6 classes in the hierarchical system shown in Table 5.3.

Table 5.3 Classes of hierarchical system

CLASS 1	
CLASS 2	
CLASS 3	
CLASS 4	
CLASS 5	
CLASS 6	

Table 5.4 Confusion matrix of network trained with flat classifier

Classes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	59	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	710	2	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	5	727	2	1	4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	417	0	6	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	2	1	0	646	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	18	30	9	616	17	5	19	3	4	0	2	2	1	2	1	11	0	3	0	0	1	0	2	5	0	0	0	1	0	2	0	0	0	6	1	0	10	1	0	0	1
7	0	1	0	0	0	0	121	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
8	0	0	1	1	0	0	0	441	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	2	2	428	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	1	0	0	475	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
11	0	0	0	0	0	0	0	0	0	1	650	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	407	0	0	0	0	0	2	2	1	0	0	1	0	0	3	4	3	0	0	8	2	0	0	0	0	0	0	10	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	684	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	714	0	0	0	10	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1	0	0	3	0	0	208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	148	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	324	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	362	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
21	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	90	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	112	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	8	0	0	0	149	0	0	0	0	0	0	27	0	0	0	0	0	0	0	1	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	460	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	8	175	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	0	0	0	0	2	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	144	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	86	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	1	0	0	0	3	0	0	0	0	0	0	1	103	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	2	0	0	0	1	1	0	2	0	0	0	0	2	266	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	1	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210	0	1	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	120	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	382	0	1	2	1	3	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	118	0	0	0	1	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0
39	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	663	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	87	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	1	83	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	6
43	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	82

Accuracy rate and training performance are presented in Table 5.5. It can be seen that the performance of this system is lower than the proposed method of the study.

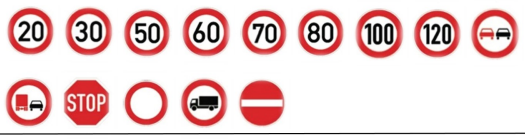
Table 5.5 Accuracy and training performance

Class	Accuracy	Training time
Main classifier(6 groups)	97.70%	35min
Group 1 (2 classes)	99.87%	3min
Group 2 (10 classes)	94.34%	15min
Group 3 (4 classes)	99.41%	7min
Group 4 (11 classes)	99.10%	17min
Group 5 (4 classes)	97.85%	8min
Group 6 (12 classes)	98.89%	19min
General structure	96.62%	104min

5.1.5 Proposed Hierarchical Classification

In the proposed hierarchical structure , SSIM and linkage-clustering has been taken advantage of. First of all similarity index matrix has been calculated between representative images of classes. This is important because these images are the most accurate form trusted to be compared. So the generated matrix is given to linkage function and it indicates that there are 4 main groups to be trained as first step of the model and then each group will be considered by its own network as shown in Table 5.6. With this system, there will be 5 network to train; one for main classifier and four for subclassifiers.

Table 5.6 Hierarchical classification by linkage clustering

FAMILY 1	
FAMILY 2	
FAMILY 3	
FAMILY 4	

5.1.6 Hyperparameters Comparison of Hierarchical Classification

There are couple of parameters like learning rate, batch size, momentum which may affect both training and test performance of the system. One thing to say about comparisons is that all the networks uses adaptive learning rate. The main learning rates are the starting point so as to be decreased through epochs because there is no need to use the same learning rate while epoch number is getting larger. The reason of that is the loss function is reaching its final point so decreasing learning rate helps the system find minimum value of loss function making small steps. In order to find the optimum parameter values for proposed system, all of them are considered in different networks. The results of these networks will be given in this section.

5.1.6.1 Learning Rate Comparison

Learning rate is a factor which affects the convergence of stochastic gradient descent algorithm. It actually means how many steps should be for each iteration to update system parameters when realizing backpropagation. Figure 5.4 to Figure 5.6 shows the accuracy and loss rate through training and validation and Table 5.8 to Table 5.10 presents the test performance. It can be seen that the optimum learning rate is chosen as 0.01 and the critic point here is that if the learning rate is chosen as 0.1 it is not to possible to reach the best solution because of unstable and infinite convergence.

5.1.6.2 Batch Size Comparison

Epoch is the number of updating cycle for the whole data to train network. In order to finish an epoch, all the data has to be processed once. Batch size is an important point because it is the amount of data for the system to update its parameters in each epoch. If batch size is chosen as 1, updating iteration is done for each sample of data which means there will be iteration number as same as data number which causes system to be trained with huge processing time. As it can be seen from Table 5.11 to Table 5.13 small amount of batch size causes more accuracy rate and training time because the

number of data considered to be processed for each iteration is little so it means more updating steps for the same amount in each epoch. The accuracy and loss of training and validation process is shown in from Figure 5.7 to Figure 5.9.

5.1.6.3 Momentum Comparison

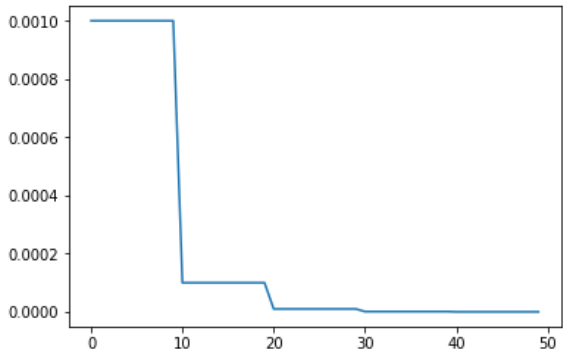
Momentum is also another crucial parameter which helps system converge faster. Without momentum, each iteration is done with only parameter which is learning rate and no previous iterations are considered as updating step. The essence of momentum is that previous iterations are taken advantage of so the convergence direction is obtained and this makes convergence faster. From Figure 5.10 to Figure 5.12 shows the effect of momentum on training and validation performance. Test performance of these networks are presented in Table 5.14 to Table 5.16.

5.1.6.4 Hierarchical Classification with Batch Normalization and Dropout

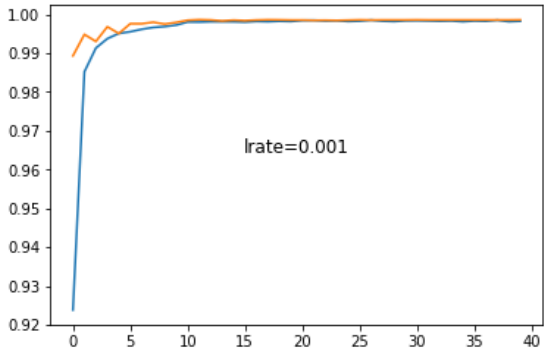
There are lots of experiment about how batch normalization and dropout effects the model in terms of both accuracy and training time. It has been wondered that the results of the network where combinations of batch normalization and dropout are applied to 4-class hierarchical model created with linkage function. Results are shown in Table 5.7.

Table 5.7 Batch normalization vs dropout

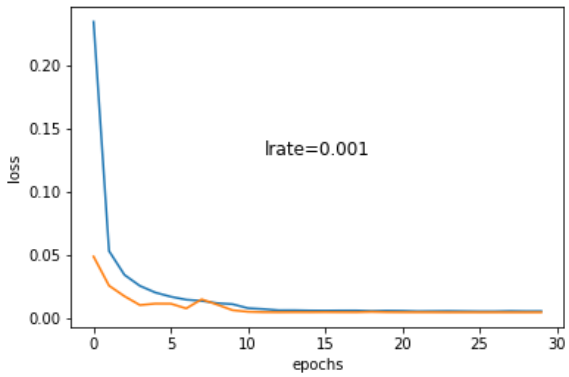
Batch Normalization	Dropout	RESULT
+	-	94.9%
-	+	97.4%
+	+	96.6%



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

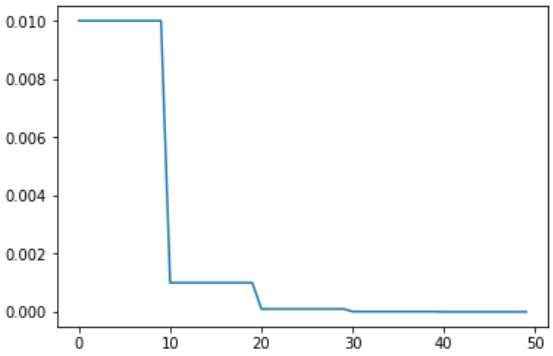


(c) Loss for training(*blue*) and validation(*red*)

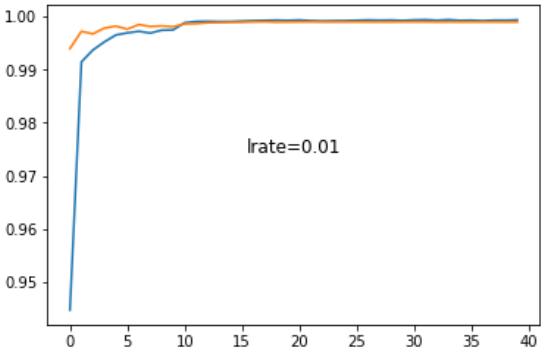
Figure 5.4 Training and validation performance - learning rate: 0.001, batch size: 32, momentum: 0.9

Table 5.8 Test performance - learning rate: 0.001, batch size: 32, momentum: 0.9

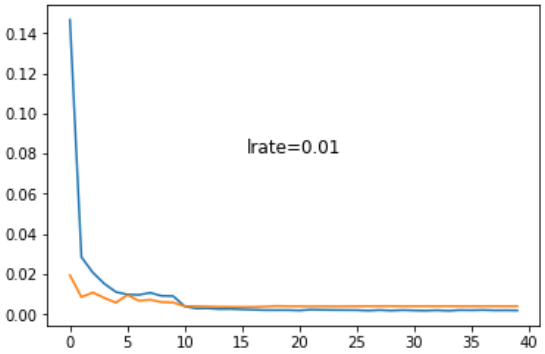
Class	Accuracy	Training time
Main classifier(4 groups)	99.47%	43min
Group 1 (14 classes)	96.98%	21min
Group 2 (6 classes)	95.53%	9min
Group 3 (8 classes)	96.78%	12min
Group 4 (15 classes)	90.90%	22min
General structure	95.20%	107min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

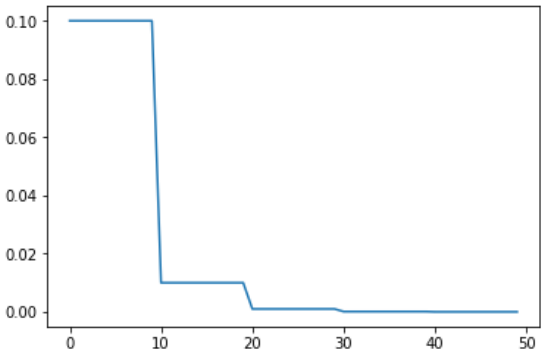


(c) Loss for training(*blue*) and validation(*red*)

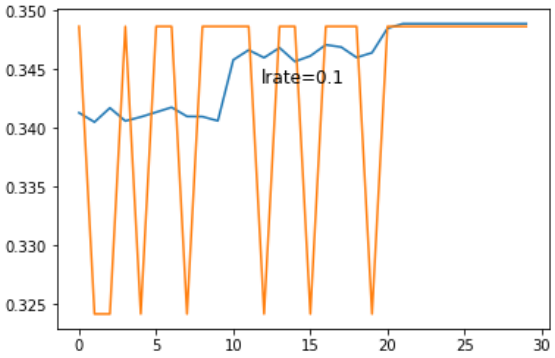
Figure 5.5 Training and validation performance - learning rate: 0.01, batch size: 32, momentum: 0.9

Table 5.9 Test performance - learning rate: 0.01, batch size: 32, momentum: 0.9

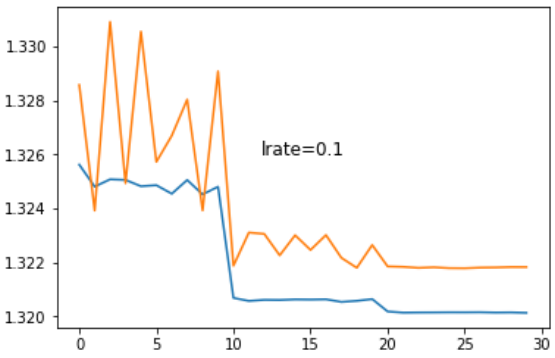
Class	Accuracy	Training time
Main classifier(4 groups)	99.46%	43min
Group 1 (14 classes)	98.35%	22min
Group 2 (6 classes)	97.90%	10min
Group 3 (8 classes)	99.15%	12min
Group 4 (15 classes)	95.62%	23min
General structure	97.42%	110min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

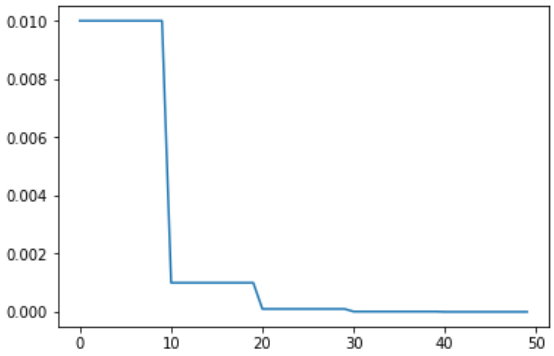


(c) Loss for training(*blue*) and validation(*red*)

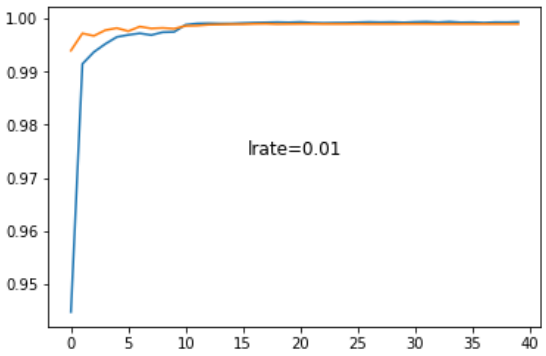
Figure 5.6 Training and validation performance - learning rate: 0.1, batch size: 32, momentum: 0.9

Table 5.10 Test performance - learning rate: 0.1, batch size: 32, momentum: 0.9

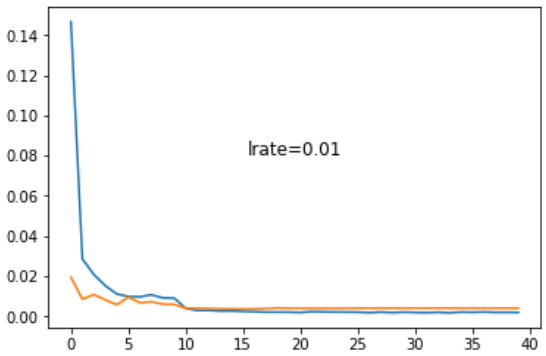
Class	Accuracy	Training time
Main classifier(4 groups)	22.10%	41min
Group 1 (14 classes)	2.38%	21min
Group 2 (6 classes)	40.67%	10min
Group 3 (8 classes)	22.03%	12min
Group 4 (15 classes)	4.30%	22min
General structure	1%	106min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

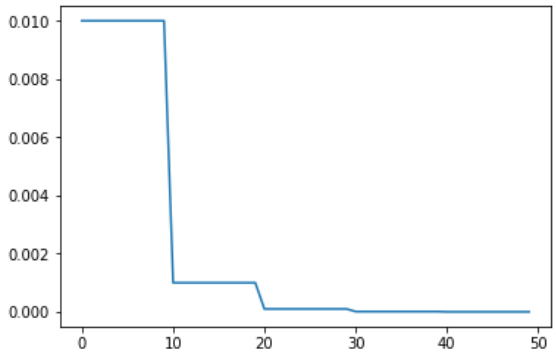


(c) Loss for training(*blue*) and validation(*red*)

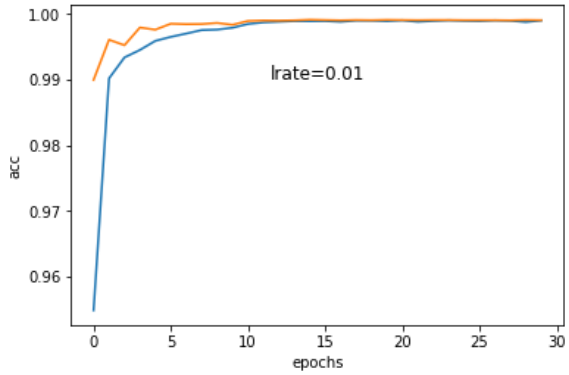
Figure 5.7 Training and validation performance - learning rate: 0.01, batch size: 32, momentum: 0.9

Table 5.11 Test performance - learning rate: 0.01, batch size: 32, momentum: 0.9

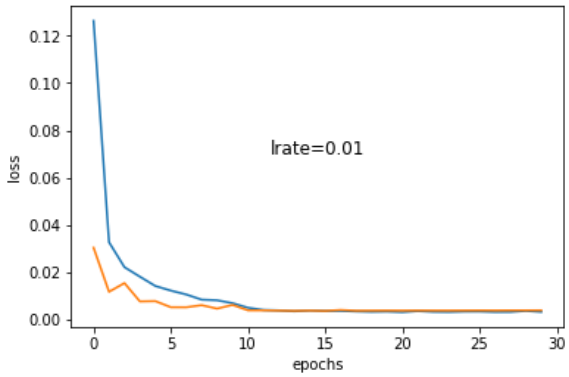
Class	Accuracy	Training time
Main classifier(4 groups)	99.46%	43min
Group 1 (14 classes)	98.35%	22min
Group 2 (6 classes)	97.90%	10min
Group 3 (8 classes)	99.15%	12min
Group 4 (15 classes)	95.62%	23min
General structure	97.42%	110min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

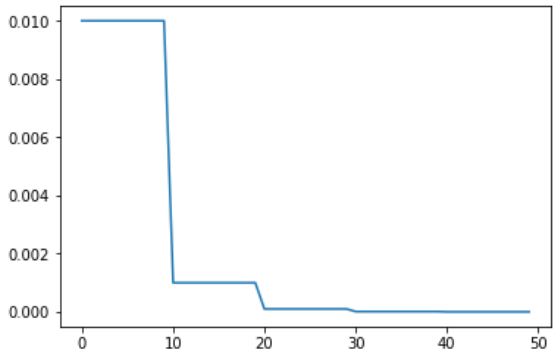


(c) Loss for training(*blue*) and validation(*red*)

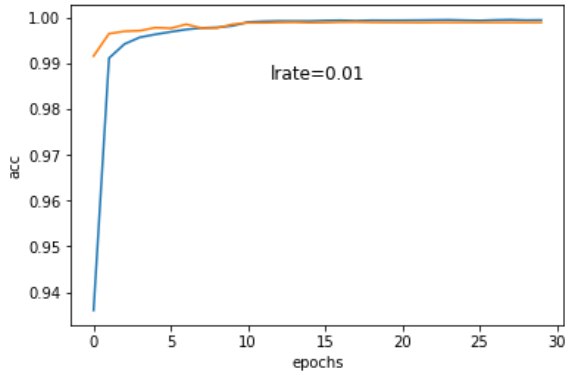
Figure 5.8 Training and validation performance - learning rate: 0.01, batch size: 64, momentum: 0.9

Table 5.12 Test performance - learning rate: 0.01, batch size: 64, momentum: 0.9

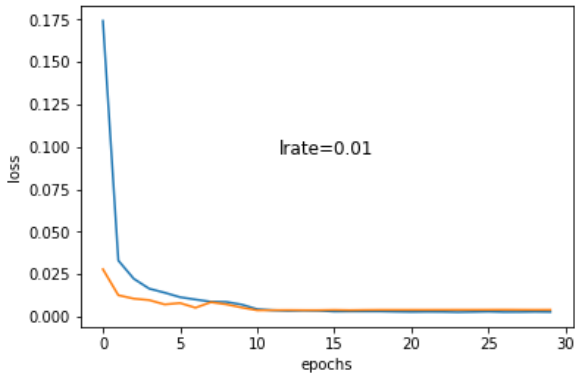
Class	Accuracy	Training time
Main classifier(4 groups)	99.36%	33min
Group 1 (14 classes)	97.79%	18min
Group 2 (6 classes)	96.04%	7.5min
Group 3 (8 classes)	98.19%	10min
Group 4 (15 classes)	95.19%	16min
General structure	96.82%	84.5min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

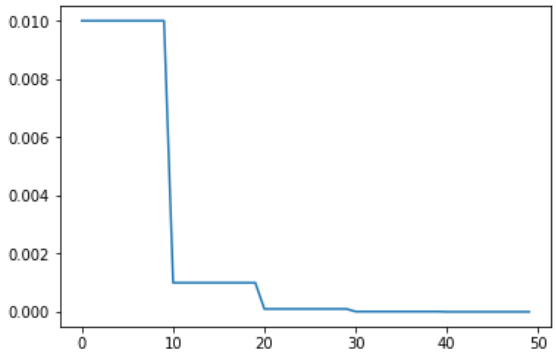


(c) Loss for training(*blue*) and validation(*red*)

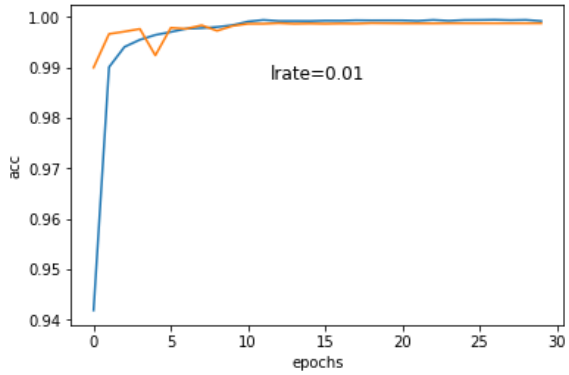
Figure 5.9 Training and validation performance - learning rate: 0.01, batch size: 128, momentum: 0.9

Table 5.13 Test performance - learning rate: 0.01, batch size: 128, momentum: 0.9

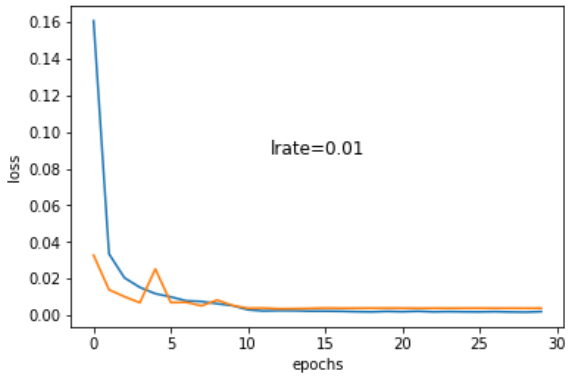
Class	Accuracy	Training time
Main classifier(4 groups)	99.52%	27min
Group 1 (14 classes)	97.47%	15min
Group 2 (6 classes)	95.53%	6.5min
Group 3 (8 classes)	97.74%	8.5min
Group 4 (15 classes)	94.51%	16min
General structure	96.40%	73min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

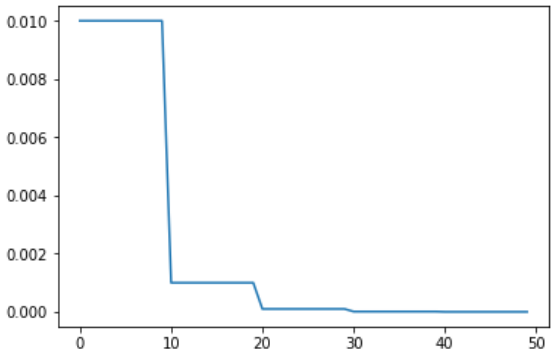


(c) Loss for training(*blue*) and validation(*red*)

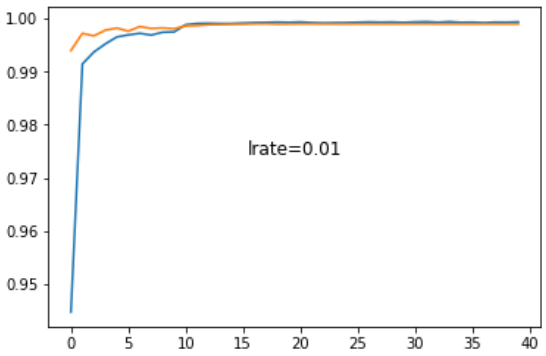
Figure 5.10 Training and validation performance - learning rate: 0.01, batch size: 32, momentum: 0.5

Table 5.14 Test performance - learning rate: 0.01, batch size: 32, momentum: 0.5

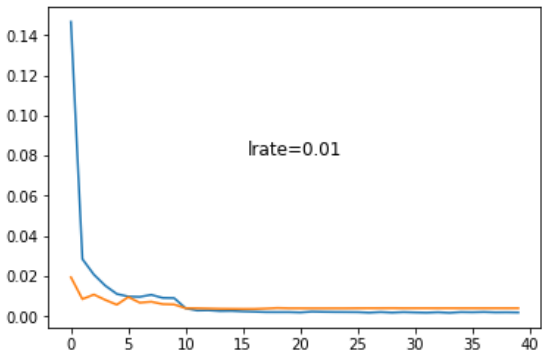
Class	Accuracy	Training time
Main classifier(4 groups)	99.43%	37.5min
Group 1 (14 classes)	97.19%	20min
Group 2 (6 classes)	95.70%	9min
Group 3 (8 classes)	97.28%	11.5min
Group 4 (15 classes)	91.64%	22min
General structure	95.56%	100min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)

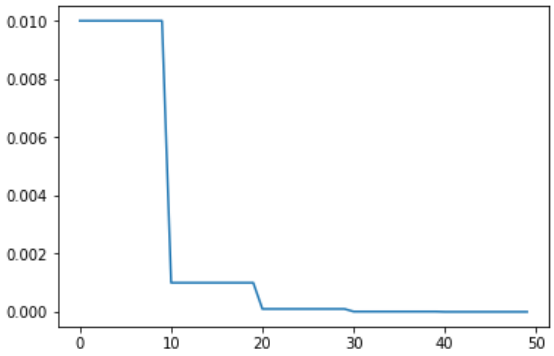


(c) Loss for training(*blue*) and validation(*red*)

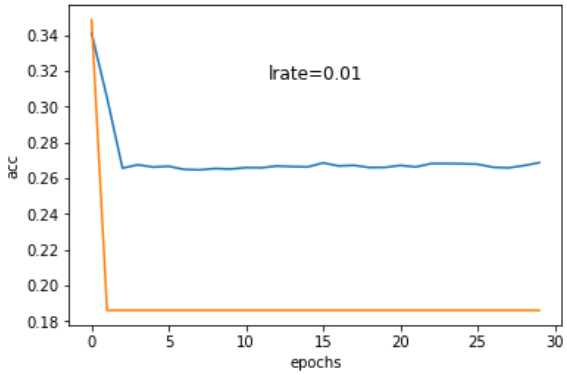
Figure 5.11 Training and validation performance - learning rate: 0.01, batch size: 32, momentum: 0.9

Table 5.15 Test performance - learning rate: 0.01, batch size: 32, momentum: 0.9

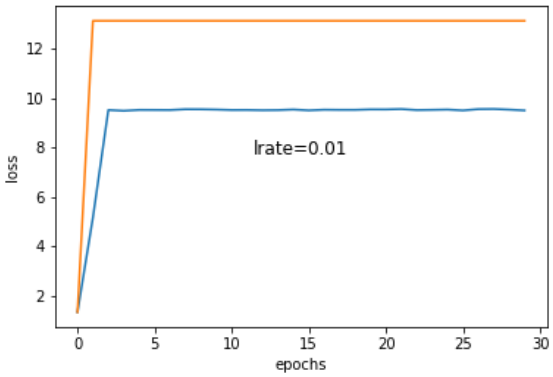
Class	Accuracy	Training time
Main classifier(4 groups)	99.46%	43min
Group 1 (14 classes)	98.35%	22min
Group 2 (6 classes)	97.90%	10min
Group 3 (8 classes)	99.15%	12min
Group 4 (15 classes)	95.62%	23min
General structure	97.42%	110min



(a) Learning rate vs epochs



(b) Accuracies for training(*blue*) and validation(*red*)



(c) Loss for training(*blue*) and validation(*red*)

Figure 5.12 Training and validation performance - learning rate: 0.01, batch size: 32, momentum: 0.99

Table 5.16 Test performance - learning rate: 0.01, batch size: 32, momentum: 0.99

Class	Accuracy	Training time
Main classifier(4 groups)	14.01%	37min
Group 1 (14 classes)	2.38%	20min
Group 2 (6 classes)	40.67%	9min
Group 3 (8 classes)	3.39%	12min
Group 4 (15 classes)	4.30%	22min
General structure	0.47%	100min

As it can be seen from the comparisons, the best solution including these parameters is the case where learning rate is 0.01, batch size is 32 and momentum is 0.9. The confusion matrix shown in Figure 5.18 for the network trained with these values belongs to the best performance which is 97.42% and it is achieved with the proposed method of the study.

5.1.7 Hierarchical Classification with Activation Function Comparison

Different activation functions can be chosen considering the fundamentals of the study. Three activation functions are tried at the output of convolution operation and results are presented in Table 5.17.

Table 5.17 Activation functions comparison

Function	RESULT
ReLU	97.4%
Hyperbolic Tangent	94.66%
Sigmoid	15.1%

As seen from Table 5.7 and Table 5.17 the best performance is achieved with dropout technique and ReLU activation function. Total training time takes approximately 2.5 hours and the accuracy is 97.4%.

5.1.8 Flat Classification with Capsule Network

One of the most recent algorithm of Deep Learning is Capsule Networks. To see how Capsule Networks are capable of classification compared to thesis study, it is trained with main 43 class. The accuracy is 96.88% and training time is beyond of all networks which is 11 hours. The study (Kumar, 2018) done with capsule network for traffic sign recognition achieved 97.6% and it is a better performance than proposed study.

Table 5.18 Confusion matrix for the best performance

Classes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	59	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	715	2	1	5	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	731	6	3	4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	4	414	0	5	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	638	2	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	12	27	5	614	3	4	9	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	105	0	0	0	0	0	0	1	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	1	1	0	442	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	2	0	0	0	437	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	1	0	0	0	0	480	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	653	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	406	0	0	0	0	0	0	3	0	0	0	0	0	0	3	0	2	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	666	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	3	0	0	0	0	0	19	715	0	2	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	270	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	2	0	1	1	1	0	1	0	0	0	0	0	1	0	206	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	148	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	324	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	350	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	1	0	90	0	1	0	0	6	0	0	0	0	6	5	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	89	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	115	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	148	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	465	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	20	0	0	0	0	0	0	1	179	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	3	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	147	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8	1	0	0	3	70	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	2	0	0	0	2	0	0	0	0	0	0	1	119	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	1	0	1	0	0	0	0	0	0	2	265	0	0	0	0	0	0	0	0	0	0	0
33	0	0	1	0	0	0	0	0	0	0	0	0	2	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	209	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	120	0	0	0	18	0	5	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	383	0	0	1	0	1	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	120	0	3	1	1	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	1	0	0	0
39	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	661	0	1	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	82	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	5
43	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	84

5.1.9 Comparison with Other Works

The proposed network is compared with the different networks proposed in the literature and comparison results are pointed out in Table 5.19.

Table 5.19 Comparison with other studies

TEAM	METHOD	RESULT
Cadence[Mao et al., 2016]	HCNN	99.67%
IDSIA[Ciregan et al., 2012]	Committee of CNNs	99.46%
COSFIRE[Gecer et al., 2017]	Color-blob-based COSFIRE filters for object recognition	98.97%
INI-RTCV[Stallkamp et al., 2012]	Human Performance	98.84%
Sermanet[Sermanet and Lecun, 2011]	Multi-Scale CNNs	98.31%
Ours	HCNN	97.4%
CAOR[Zaklouta et al., 2011]	Random Forests	96.14%
INI-RTCV[Stallkamp et al., 2012]	LDA on HOG 2	95.68%
INI-RTCV[Stallkamp et al., 2012]	LDA on HOG 1	93.18%
INI-RTCV[Stallkamp et al., 2012]	LDA on HOG 3	92.34%

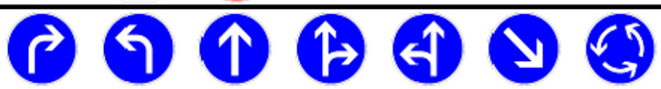
5.1.10 Network Comparison with Same Dataset

Every study compared with this study mainly preferred GTSRB dataset but each of them uses different kind of augmentation techniques for the original dataset and trains their system with augmented ones. So to have a fair comparison the dataset is applied on the studies mentioned in literature and results are shown in Table 5.20 and it shows how proposed network performs better at the same dataset usage. Classes created for the study (Mao et al., 2016) is shown in Table 5.21.

Table 5.20 Network comparison with same dataset

TEAM	METHOD	RESULT
Proposed Network	HCNN	97.4%
Cadence[Mao et al., 2016]	HCNN	97.27%
Sermanet[Sermanet and Lecun, 2011]	Multi-Scale CNNs	96.11%

Table 5.21 Families grouped in (Mao et al., 2016)

FAMILY 1	
FAMILY 2	
FAMILY 3	
FAMILY 4	
FAMILY 5	

5.2 Visualization of HCNN

When talking about CNN, there is always a term consistently used named ”black box”. Features are extracted by itself in CNN and sometimes it may be so hard to clarify why the network acts like this. So next figures will be helpful in order to visualize the feedforward propagation and see the feature maps after each convolutional layer.

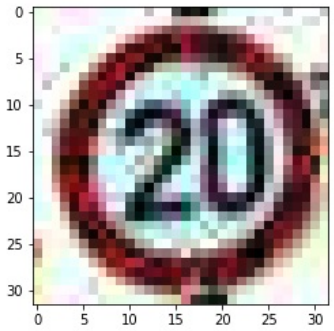


Figure 5.13 Input image for HCNN (32x32x3)

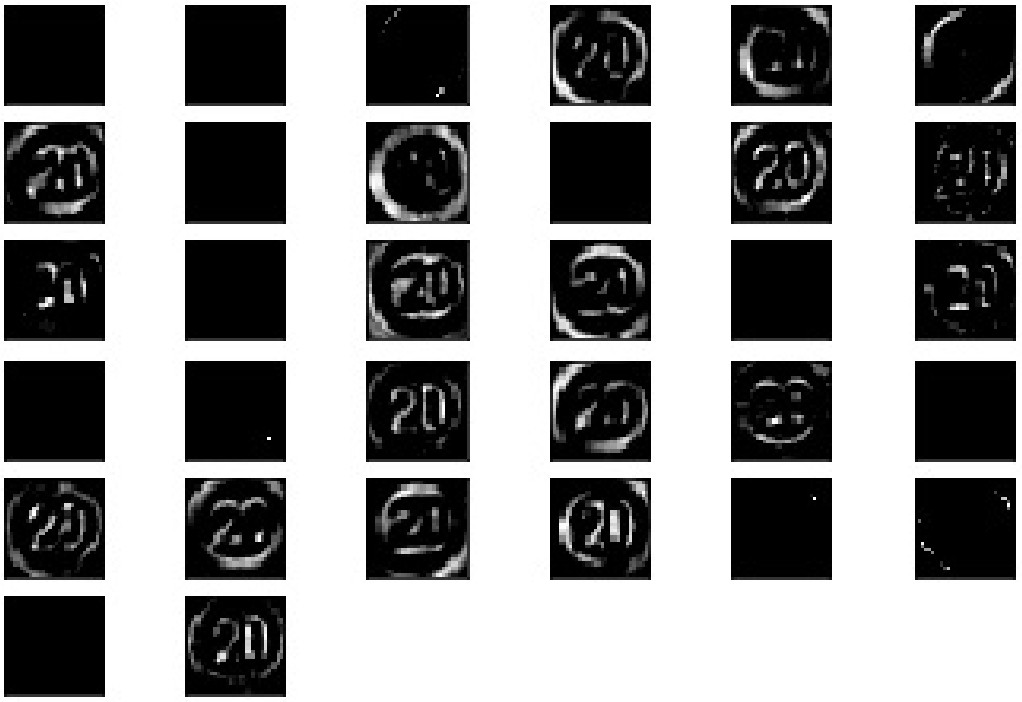


Figure 5.14 Feature maps after 1st convolution layer (28x28x32)

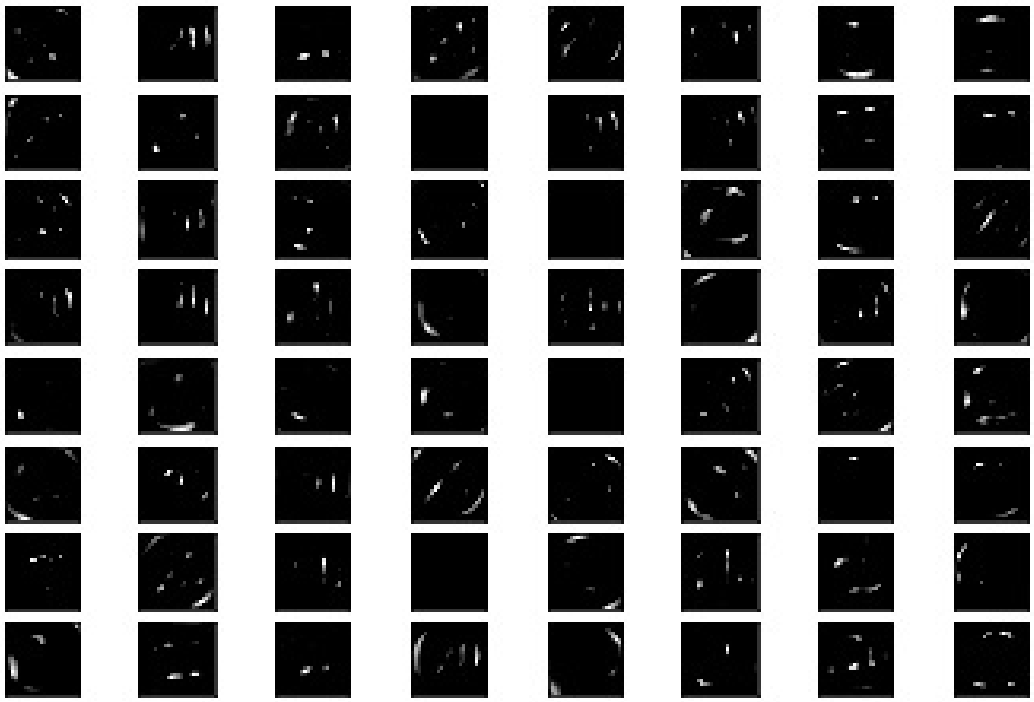


Figure 5.15 Feature maps after 2nd convolution layer (24x24x64)

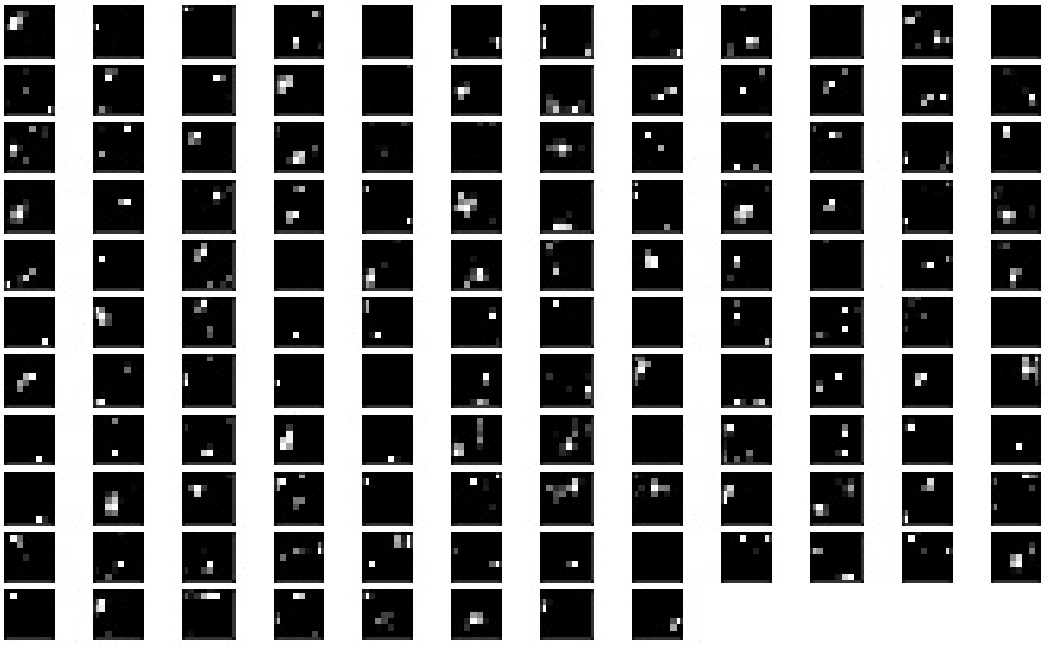


Figure 5.16 Feature maps after 3rd convolution layer (8x8x128)

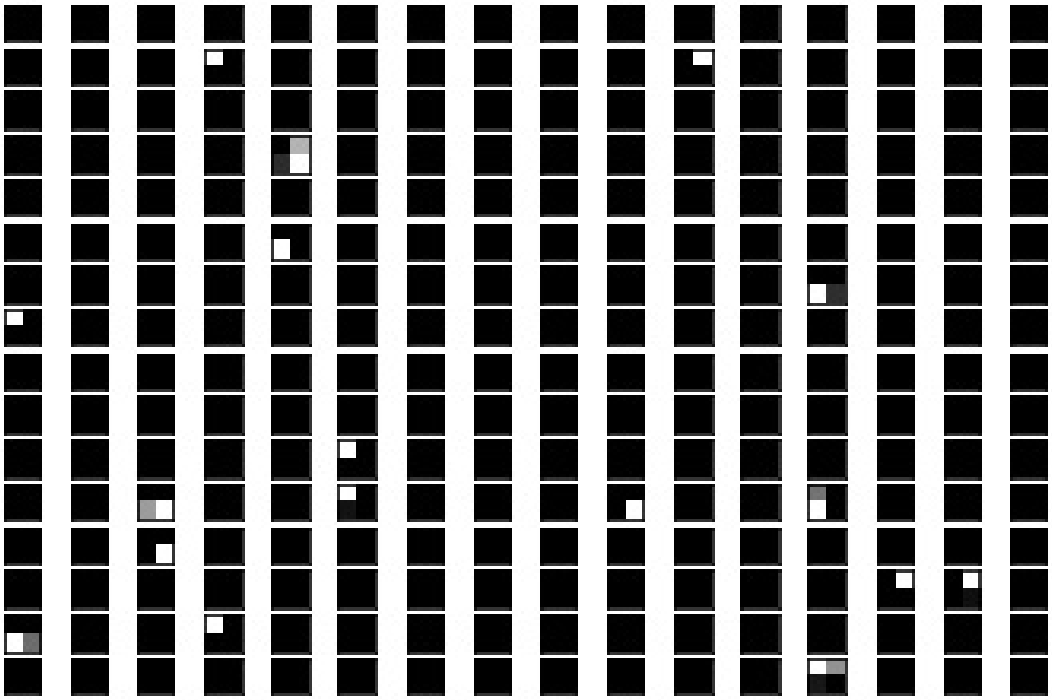


Figure 5.17 Feature maps after 4th convolution layer (2x2x256)

And the last step is the output layer which will be fired with one-hot encoded system so this image belongs to the first class of network which is correctly classified for the main divider part of the study.

Output : [[1.0000000e+00, 2.4924084e-31, 2.0833452e-29, 4.2591074e-28]]

CHAPTER SIX
CONCLUSION

The interpretation of traffic signs and the situations of the road are no longer to be a difficulty for people while they are driving because they are now processed with machine learning and evaluated by AI. Since the human life matters, traffic sign recognition systems are required to be error-free. For this reason, getting the highest performance in traffic sign recognition systems is not only necessary but also mandatory. In this thesis, to increase the sign recognition performance a hierarchical classifier is proposed since the data set of traffic signs is categorical.

The proposed structure consists of five classifiers. The first level classifier divides the data into four groups, while the lower level classifiers do in-group classification. CNN's are chosen as classifiers. There are some considerable outputs of CNN which can be extracted from the study. The first one is the number of weight parameters are much more less than a standard MLP structure thanks to weight-sharing property of CNN so training is more faster than MLP having the same dimension of layer. In general meaning CNN works as a feature extractor using convolution operation but how to do is a black box for now. CNN is a deep model of MLP and convolution is the main operation in this structure. Pooling operation is useful for preventing system to be overfitting but on the other hand it causes a part of data vanishing at the end of the layer. And if dropout rate is kept as low value the system also tends to be overfitted. It has been observed that 0.5 is ideal as dropout rate for convolution layer. As dropout there are some hyperparameters for backpropagation of the system which is related to stochastic gradient descent algorithm. The effect of learning rate, batch size and momentum values has been examined and it has been figured out that these parameters are important to reach the optimum solution for the network. Activation function used in convolution is also important. As seen in the results some functions may cause stuck in training process because it is not appropriate to be used for the study when considering outputs of each convolution layer. Selection of filters also plays a considerable role in determining how much coarse feature is extracted. If the size of filters kept small the

approach is towards fine-grained extraction but if it is kept big coarse-grained extraction is achieved. Of course there are some truths which is bounded to the hardware and dataset part of the study. For example, with an increasing GPU speeds, it will be much more easier to implement a CNN algorithm since GPUs are so appropriate for matrix calculation and CNN works efficiently when the dataset is getting larger.

The result obtained for hierarchical structure shows that if a CNN would have deeper layers accuracies will be much more higher as training time will get longer . All the trials whether it is going to be successful or not when comparing to other studies have been done and the consequences have been given at the application and result section. The performance rate 97.4% is obtained with the proposed hierarchical structure created with the standard CNN. Based on the data, It can be said that it gives better results than the other studies, but the importance of the training set created through preprocessing should not be ignored.

As a conclusion,there is no doubt that machine learning is a milestone for taking classical classification to the next level and it is guaranteed that there will be plenty of studies will be published in this area. The study shows that machine learning methods are capable of achieving different kind of problems no matter how much difficult it is and it encourages people to deal with the problems with machine learning while there is a continuous improvement on this area.

REFERENCES

Ahn, S. H. (2019). *Convolution*. Retrieved August 25, 2019, from http://www.songho.ca/dsp/convolution/convolution.html.

Bodington, D., Greenstein, E. & Hu, M. (2015). Implementing machine learning algorithms on GPUs for real-time traffic sign classification. Retrieved May 10, 2019, from http://cs229.stanford.edu/proj2014

Brkic, K. (2010). An overview of traffic sign detection methods. *Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska*, *3*, 10000.

Brownlee, J. (2019). *A tour of the most popular machine learning algorithms*. Retrieved August 20, 2019, from https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.

Brownlee, J. (2016). *Crash course on multi-layer perceptron neural networks*. Retrieved August 21, 2019, from http://machinelearningmastery.com/neural-networks-crash-course/.

Brownlee, J. (2019). *Machine learning categories*. Retrieved August 12, 2019, from https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.

Caraffi, C., Cardarelli, E., Medici, P., Porta, P.P., Ghisio, G. & Monchiero, G. (2008). An algorithm for Italian de-restriction signs detection. *2008 IEEE Intelligent Vehicles Symposium*, 834-840.

Ciresan, D.C., Meier, U. & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3642-3649.

dos Santos, L. (n.d.). *Convolution*. Retrieved August 14, 2019, from https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html.

Ellahyani, A., Ansari, M. E. & Jaafari, I. E. (2016). Traffic sign detection and recognition based on random forests. *Applied Soft Computing, 46,* 805-815.

Filkovic, I. (2014). Traffic sign localization and classification methods : An overview. *Computer Vision Innovation for Safe Traffic*, 1-10.

GTSRB (2013). *GTSRB dataset annotations.* Retrieved August 15, 2019, from http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset.

Godbole, S. (2002). Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers. *Annual Progress Report, March 2002,* 17.

Haloi, M. (2016). Traffic sign classification using deep inception based convolutional networks. *ArXiv,* abs/1511.02992, 1-5.

Haloi, M. (2015). A novel plsa based traffic signs classification system. *ArXiv,* abs/1503.06643, 1-5.

Hasasneh, N., Daraghmi, Y. A. & Hasaneh, A. (2016). Accurate real-time traffic sign recognition based on the connected component labeling and the color histogram algorithms. *International Journal of Signal Processing Systems, 4*(5), 417-421.

He, X. Z., Ren, S. & Sun, J. (2015). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 770-778.

Hijazi, S., Kumar, R. & Rowen, C. (2015). Using convolutional neural networks for image recognition. Retrieved August 15, 2019, from https://www.embedded-vision.com/platinum-members/cadence/embedded-vision-training/documents/pages/neuralnetworksimagerecognition.

Hossain, M. S. & Hyder, Z. (2015). Traffic road sign detection and recognition for automotive vehicles. *International Journal of Computer Applications*, *120*(24).

Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 1-11

Kafunah, J. (2016). *Backpropagation in convolutional neural networks*. Retrieved August 10, 2019, from http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/.

Karpathy, A. (n.d.). *Convolutional neural networks (CNNs / ConvNets)*. Retrieved August 16, 2019, from http://cs231n.github.io/convolutional-networks/.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097-1105.

Kumar, A.D. (2018). Novel deep learning model for traffic sign detection using capsule networks. *ArXiv*, abs/1805.04424.

Li, H., Sun, F., Liu, L. & Wang, L. (2015). A novel traffic sign detection method via color segmentation and robust shape matching. *Neurocomputing*, *169*, 77-88.

Loomis, J. (n.d.). *2D Convolution*. Retrieved August 9, 2019, from http://www.johnloomis.org/ece563/notes/filter/conv/convolution.html.

Mao, X., Hijazi, S. L., Casas, R. A., Kaul, P., Kumar, R. & Rowen, C. (2016). Hierarchical cnn for traffic sign recognition. *2016 IEEE Intelligent Vehicles Symposium (IV)*, 130-135.

MATLAB (2016). *version 9.0.0.341360 (R2016a)*. The MathWorks Inc., Natick, Massachusetts.

Møgelmose, A., Trivedi, M. M. & Moeslund, T. B. (2012). Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, *13*, 1484-1497.

Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378, 1-20.

Pavlov, N. (n.d.). *MLP neural net*. Retrieved August 1, 2019, from https://github.com/nikolaypavlov/MLPNeuralNet.

Qian, R., Yue, Y., Coenen, F. & Zhang, B. (2016). Traffic sign recognition with convolutional neural network based on max pooling positions. *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 578-582.

Rimal, K. (2018). *Capsule network*. Retrieved August 11, 2019, from https://software.intel.com/en-us/articles/understanding-capsule-network-architecture.

Sayad, S. (n.d.). *Linkage clustering*. Retrieved October 18, 2019, from https://www.saedsayad.com/clustering_hierarchical.htm.

Sermanet, P. & Lecun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of the International Joint Conference on Neural Networks*, 2809-2813.

Shirani, H. (2015). Applications of deep learning to sentiment analysis of movie reviews. Retrieved June 16, 2019, from https://cs224d.stanford.edu/reports/Shirani-MehrH.pdf.

Stallkamp, J., Schlipsing, M., Salmen, J. & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32, 323-332.

Surucu, E. A. & Doğan, H. (2018). Traffic sign recognition with hierarchical convolutional neural network. *26th Signal Processing and Communications Applications Conference (SIU 2018)*, 1-4.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2014). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9.

Wang, Z. (2015). The applications of deep learning on traffic identification. *BlackHat USA*, 24.

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*, 600-612.

Wen, L. H. & Jo, K. H. (2017). Traffic sign recognition and classification with modified residual networks. *2017 IEEE/SICE International Symposium on System Integration (SII)*, 835-840.

Wilson, R. (2015). *Can you see using convolutional neural networks?* Retrieved August 11, 2019, from http://systemdesign.altera.com/can-you-see-using-convolutional-neural-networks/.

Wu, Y., Liu, Y., Li, J., Liu, H. & Hu, X. (2013). Traffic sign detection based on convolutional neural networks. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1-7.

Yaldex (n.d.). *Backpropagation*. Retrieved August 15, 2019, from http://www.yaldex.com/game-development/1592730043_ch20lev1sec5.html.

Yin, S., Ouyang, P., Liu, L., Guo, Y. & Wei, S. (2015). Fast traffic sign recognition with a rotation invariant binary pattern based feature. *Sensors*, *15*(1), 2161-2180.

Youssef, A., Albani, D., Nardi, D. & Bloisi, D. D. (2016). Fast traffic sign recognition using color segmentation and deep convolutional networks. *International Conference on Advanced Concepts for Intelligent Vision Systems*, 205-216.

Ma, Y. & Huang, L. (2015). Hierarchical traffic sign recognition based on multi-feature and multi-classifier fusion. *First International Conference on Information Science and Electronic Technology (ISET 2015)*.

Zadeh, R. B. (2018). *TensorFlow for deep learning*. California: O'Reilly Media, Inc.

Zakir, U., Leonce, A. N. J. & Edirisinghe, E. A. (2010). Road sign segmentation based on colour spaces: A comparative study. *Proceedings of the 11th IASTED International Conference on Computer Graphics and Imaging, CGIM 2010*, 72-79.

Zeng, Y., Xu, X., Fang, Y. & Zhao, K. (2015). Traffic sign recognition using extreme learning classifier with deep convolutional features. *The 2015 International Conference on Intelligence Science and Big Data Engineering (IScIDE 2015), 9242*, 272-280.

Zhou, J., Zhao, J., Zhang, Z. & Li, J. (n.d.). *Capsule vs CNN*. Retrieved August 15, 2019, from https://docs.dgl.ai/en/latest/tutorials/models/4_old_wines/2_capsule.html.

Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B. & Hu, S. (2016). Traffic-sign detection and classification in the wild.*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2110-2118.

Zhu, Y., Zhang, C., Zhou, D., Wang, X., Bai, X. & Liu, W. (2016). Traffic sign detection and recognition using fully convolutional network guided proposals. *Neurocomputing, 214*,758-766.