MARCH 2021 M.Sc. in Electrical and Electronics Engineering BÜNYAMİN DİKİCİ

REPUBLIC OF TURKEY

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

OBJECT DETECTION USING CONVOLUTIONAL NEURAL NETWORK

M.Sc. THESIS

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

BY

BÜNYAMİN DİKİCİ

APRIL 2021

OBJECT DETECTION USING CONVOLUTIONAL NEURAL NETWORK

M.Sc. Thesis

in

Electrical and Electronic Engineering

Gaziantep University

Supervisor

Asst. Prof. Dr. Serkan ÖZBAY

by

Bünyamin DİKİCİ

April 2021

©2021[Bünyamin DİKİCİ]

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Bünyamin DİKİCİ

ABSTRACT

OBJECT DETECTION USING CONVOLUTIONAL NEURAL NETWORK

DİKİCİ, Bünyamin
M.Sc. in Electrical and Electronics Engineering
Supervisor: Asst. Prof. Dr. Serkan ÖZBAY
April 2021
84 pages

Increasing amounts of digital images are used in many applications around the world and this usage rate is constantly increasing. There are cameras as sensors to capture images for different purposes especially for automation purposes. Object detection is a technique for identifying the objects in images. The main goal is to understand what an image involves. Object detection is still challenging task in computer vision related fields and that with the recent developments in deep neural networks, recognizing the objects in images is now possible with higher accurate rates. In this study, Keras library and convolutional neural network technique were used for object detection and the effects of some design parameters on performance were examined. These hyperparameters are batch size, train image number, epoch number, learning rate, filter numbers, activation functions, input images sizes, dropout, class number on data set and noise. The experiments were made with created 35 different design parameters and two popular deep learning architectures. After that metric results which belong to design parameters were obtained. Furthermore they were visualized by means of graphs and tables. German Traffic Sign Dataset, which contains approximately 50000 image data, was used on training and testing because of involving adequate images on the dataset.

Keywords: Convolutional Neural Network, Object Detection, Hyperparameter, Keras, German Traffic Sign Dataset

ÖZET

EVRİŞİMLİ SİNİR AĞLARI KULLANARAK NESNE BELİRLEME

DİKİCİ, Bünyamin
Yüksek Lisans Tezi, Elektrik ve Elektronik Mühendisliği Bölümü
Danışman: Dr. Öğr. Üyesi Serkan ÖZBAY
Nisan 2021
84 sayfa

Dünya genelinde birçok uygulamada giderek artan miktarda dijital görüntü kullanılmakta ve bu kullanım oranı sürekli artmaktadır. Özellikle otomasyon amaçlı olmak üzere, farklı sebeplerle görüntü yakalayan ve sensör olarak kullanılan kameralar bulunmaktadır.Nesne algılama, görüntülerdeki nesneleri tanımlamaya yönelik bir tekniktir. Temel amaç, bir görüntünün ne içerdiğini anlamaktır. Nesne algılama, bilgisayarla görme ile ilgili alanlarda hala zorlu bir görevdir ve derin sinir ağlarındaki son gelişmelerle, görüntülerdeki nesneleri tanımak artık daha yüksek doğru oranlarla mümkündür. Bu çalışmada nesne tespiti için Keraskütüphanesiileevrişimli sinir ağı tekniğini kullanıldı ve bazı tasarım parametrelerinin performans üzerindeki etkileri incelendi. Bu hiperparametreler; yığın boyutu, eğitimdeki görüntü sayısı, çevrim sayısı, öğrenme oranı, filtre sayısı, aktivasyon fonksiyonu, giriş görüntülerinin boyutları, seyreltme, veri setindeki sınıf numarası ve gürültüdür. Denemeler, oluşturulan 35 farklı tasarım parametresi ve popüler iki derin öğrenme mimarisi ile yapılmıştır. Daha sonra tasarım parametrelerine ait metrik sonuçlar elde edilmiştir. Ayrıca sonuçlar grafik ve tabloyardımıylagörselleştirilmiştir. Yaklaşık 50000 görüntü verisi içeren Alman Trafik İşareti Veri Kümesi, bünyesinde yeterli görüntü bulunması nedeniyle eğitim ve testlerde kullanılmıştır.

Anahtar Kelimeler: Evrişimli Sinir Ağları, Nesne Belirleme, Hiperparametre, Keras, Alman Trafik İşareti Veri Seti

"Dedicated to my family"

ACKNOWLEDGEMENTS

I am very grateful for the encouragement and motivation of my supervisor who showed great effort in the emergence of this master thesis work with his unique patience and experience in this long process.

I would like to express my eternal love and gratitude to my all family members, who are with me in all circumstances, for their support. Also I would like to thank my other precious promoters who appreciate and support my working with their all good energies. Always best wishes...

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

SSD	Solid State Drive
RTSD	Russian Traffic Sign Dataset
STSD	Swedish Traffic Sign Dataset
GTSD	German Traffic Sign Dataset
CNN	Convolutional Neural Network
ICANN	International Conference on Artificial Neural Networks
MNIST	Modified National Institute of Standards and Technology
RGB	Red Green Blue
ILSVRC	Image Net Large Scale Visual Recognition Challenge
DL	Deep Learning
ML	Machine Learning
AI	Artificial Intelligence
CV	Computer Vision
LSTM	Long Short Term Memory
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
HOG	Histogram of Oriented Gradients
ANNs	Artificial Neural Networks
API	Application Programming Interface
DP	Design Parameter

CHAPTER 1

INTRODUCTION

1.1 General View

“The exciting new effort to make computers think… machines with minds, in the full and literal sense.”

---- John Haugeland

This statement, uttered by J. Haugeland in 1985, showed how much we were keen on a formation that could take on our daily work instead of us and lighten our burden. A world where robots have the physical and mental abilities of humans and even make and implement decisions just like humans in daily life, thus making our lives easier, has embellished the dreams of human beings for years. In order that this dream to come true, first of all, machines equipped with artificial intelligence had to see and perceive the objects its around and be able to distinguish these objects from each other. In addition to technological devices such as phones, computers and tablets we use today, public transportation vehicles such as buses and subways, automobiles, public areas, private workplaces and streets are equipped with cameras. The use of cameras in such a wide area offers artificial intelligence to have an eye everywhere and a large amount of photo and video possibilities that can be processed. These data, which are processed by way of the algorithms created, allows vehicles with autonomous driving systems to make decisions on behalf of people in some cases. It would be quite wrong to say that the abilities of artificial intelligence are only processing image and video data. Apart from the automotive sector, it is used extensively in many fields such as health, finance and economy, textile and fashion, education, law, marketing and astronomy.

Generally, people want to share information they have learned thus other people benefit from it. Also, anyone may want to review their data later. The most efficient method for this is to save the information that needs to be retrieved later. This adventure, which started with the use of cave walls as registration points, continued with the emergence of the possibility of recording on electronic devices as the years progressed. Today, much faster and higher capacity data storage units are available for our use. Increasing of data storage areas such as cloud or Solid State Drive (SSD) and providing easier access to these areas have caused to enter our lives of the big data concept. In fact, we cause to grow unwittingly this data stack, and we even offer it to the use of large companies. As technology advances, in parallel with the increase on the number of data that can be processed, more powerful devices that can process data have started to emerge. The fact that we have more data and more powerful devices has led to the formation of machine learning and deep learning concepts, which are generally considered as a sub-discipline of artificial intelligence. Deep learning is an approach to artificial intelligence. Especially, it is a type of machine learning, a technique which provides computer systems to move forward with experience and data (Bengio et al., 2016). With these methods, we can process data faster and get more accurate results.

Deep learning, a branch of artificial intelligence, is increasing its impact on five key industry categories such as health, manufacturing, customer service, finance and transportation. For example, let's assume that we are moving with our vehicle on a road where the speed limit is 50 km/h and there are warning traffic signs. If our speed goes above 50 km/h without us being aware of it, our speed will automatically fall back to 50 km/h when the artificial intelligence detects the warning traffic sign. Thus, it will minimize the possible accident risks that may occur in these and similar situations. According to the estimation of Gartner, one of the leading organizations conducting research on Information Technology products, more than 740,000 autonomous vehicles will take their place in the market in 2023 (Rimol, 2019).

1.2 Aim of the Study

People's learning adventure continues uninterrupted throughout their lives. Especially in childhood, we begin to explore the objects around us and distinguish them from each other. When we see different types and instances of an objects many times, the easier it will be for us to perceive that object and distinguish it from others.

The same is true for deep learning method. In this discipline, which is based on self-education, the high number of pictures in the data set will have a positive effect. Therefore, the first step was taken by choosing a suitable data set for this study. Traffic signs ensure very important visual information that can be help us for proper driving conditions (Wali et al., 2015). For instance, they warn us about speed limits, roadway access, obstacles, drivable lanes, temporary situations, restrictive areas and so on (Stallkamp et al., 2012). Although there are many traffic sign data sets available, if a data set containing over 100,000 images such as the Russian Traffic Sign Dataset (RTSD) were used, much longer times would be required to obtain the results of the design parameters. If a data set containing approximately 4000 images, such as the Swedish Traffic Sign Dataset (STSD), was used, it would not be possible to generalize about the design parameters created since the number of pictures and classes would be relatively insufficient. For this reason, the German Traffic Sign Dataset (GTSD) which has about 50000 image data was chosen as the most ideal data set and used in this study. In order to use the Convolutional Neural Network (CNN) technique, the GTSD used in the International Conference on Artificial Neural Networks (ICANN) held in 2011 was used.

Convolutional neural networks have been extremity prosperous in applied applications (Bengio et al., 2016). In 1989, CNN technique with backpropagation was mentioned for the first time in the work titled Backpropagation Applied to Handwritten Zip Code Recognition by Yann LeCun et al. Later, Yann LeCun and others presented the first successful CNN architecture, Lenet-5 (Lecun, et al., 1989). With this architecture applied to the Modified National Institute of Standards and Technology (MNIST) data set, quite good results were obtained at that time. The fact that the input images in the data set have a single channel also contributed to this success. This architecture cannot show the same success in Red Green Blue (RGB) pictures. Inability to get the desired result exactly, not having enough data and the lack of powerful devices to process the data caused deep learning to be shelved for a while. Over the years, the number of image data that can be processed has increased and more powerful devices have emerged that can process this data. Thus, it has become possible to obtain more accurate results by processing more data in a shorter time with the deep learning methods. With these developments, Ilya Sutskever and Geoffrey Hinton were the winners of the Image Net Large Scale Visual Recognition Challenge (ILSVRC) with AlexNet architecture in 2012 (Krizhevsky et al., 2012).

Thanks to this success and the possibilities offered by technological developments, deep learning has become popular again. Therefore, CNN technique was preferred as main technique of this study.

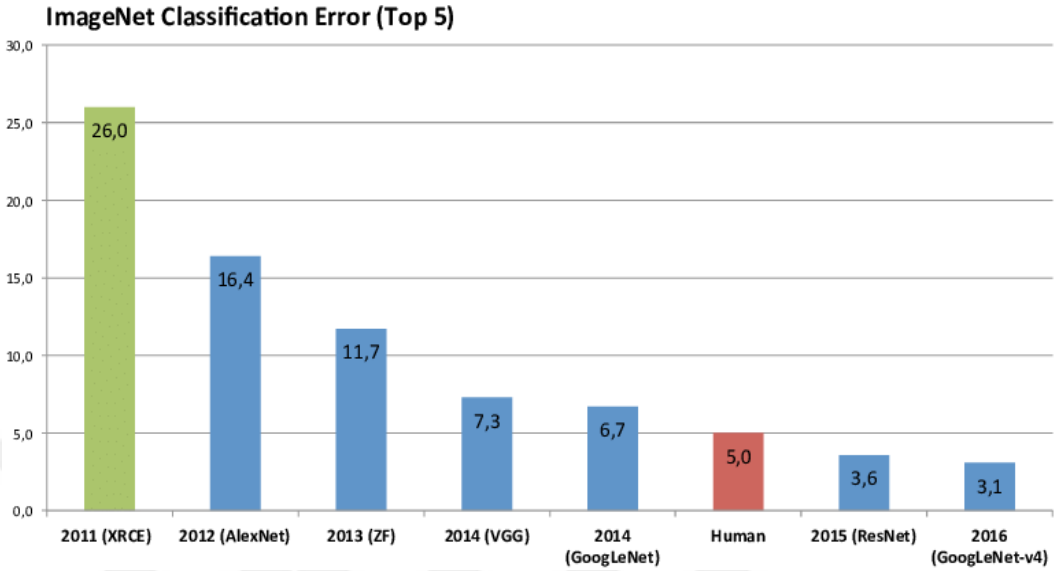

Figure 1.1 Error values of champions of ILSVRC (Padmanabhan, 2019)

Ubuntu Operating System, Python Software Language and Tensorflow based Keras Library were used while developing the design parameters. In these elections, the fact that all three are free and have quite a lot of resources has been an important criterion at the selection stage.

For the application of CNN technique, popular architectures such as AlexNet, VGG, ZFNet, GoogleNet and Microsoft ResNet can be applied or a new architecture can be created. However, in both options, the hyperparameters must be adjusted properly to achieve high accuracy. Many deep learning algorithms depend on hyperparameter choices. These selections can have great effect on system performance (Bergstra et al., 2013). Choosing the correct values when setting the hyperparameters is predictable but there are no exact values. In this case, it is necessary to benefit from past experiences and reach the optimum result by trying. The aim of this thesis isn't to get better result for object detection. The main purpose of this thesis is to observe the performances of different design parameters that correctly classify the traffic signs using CNN. Firstly, AlexNet and Lenet-5 deep learning architectures was applied on GTSD. While the results obtained from AlexNet were high, the results

obtained from LeNet-5 were low. For this reason, it was necessary to create an architecture with lower performance compared to AlexNet and higher performance compared to Lenet-5. After that, an algorithm was created randomly and then 34 different design parameters was derived by making changes on hyperparameters such as batch size, train image number, epoch number, learning rate, filter numbers, activation functions, input images sizes, dropout, noise, class number on data set. After that the effect of the changes on the result was examined with accuracy, loss, precision, recall and f-score metric parameters. Design parameter 1 is chosen as a reference algorithm. By comparing the results obtained from other design parameters with the results obtained from design parameter 1, the points to be considered in the selection of hyperparameters are revealed. It has been observed that about 98% accuracy and F-score values can be obtained as a result of using the correct hyperparameters.

1.3 Road Map

Chapter 1 clarifies general view to topics of this thesis, aim of this study and road map explaining issues mentioned in other sections. In this study, there are 5 chapters other than the chapter we are in. The subjects explained in these chapters are listed as follows:

In Chapter 2, the brief history of the CNN and deep learning are mentioned. On the other hand, related works are reviewed.

In Chapter 3, deeper information about the technical terms and topics used in the thesis is given. Thus, increase the understandability of future topics is aimed.

In Chapter 4, the changes made on the hyperparameters in the design parameters and the effect of these changes on the result were observed numerically and graphically.

In Chapter 5, the results obtained and the techniques used are evaluated and the correct hyperparameter selection is mentioned. On the other hand, the study was concluded by mentioning the situations that can be added to the study in the future and that can take the study further.

CHAPTER 2

BACKGROUND

In Chapter 2 which we are in, time line of deep learning and history of CNN were explained. Besides, a section in this chapter was dedicated to the previous related works done about of this thesis main subject.

2.1 History of Deep Learning

Ever-increasing social media user number, larger data sets, more easily accessible greater storage areas and suitable machines equipped with more powerful GPUs that can process any data faster have been the biggest driving forces for the rise of deep learning. Deep learning which have several different naming like hierarchial learning or deep machine learning is a new machine learning research area and one of the essential goals of machine learning's is to get closer to artificial intelligence (Deng and Yu, 2014).

The concepts of Deep Learning (DL), Machine Learning (ML), and Artificial Intelligence (AI) have become interchangeable and confused terms. However, it would be more appropriate to consider these concepts as disciplines that have evolved from artificial intelligence to machine learning and from machine learning to deep learning. Deep learning is often defined as an approach or sub-discipline of machine learning. Similarly, machine learning is considered as a sub-discipline of artificial intelligence. For this reason, when researching the history of deep learning, the starting point should be artificial intelligence.

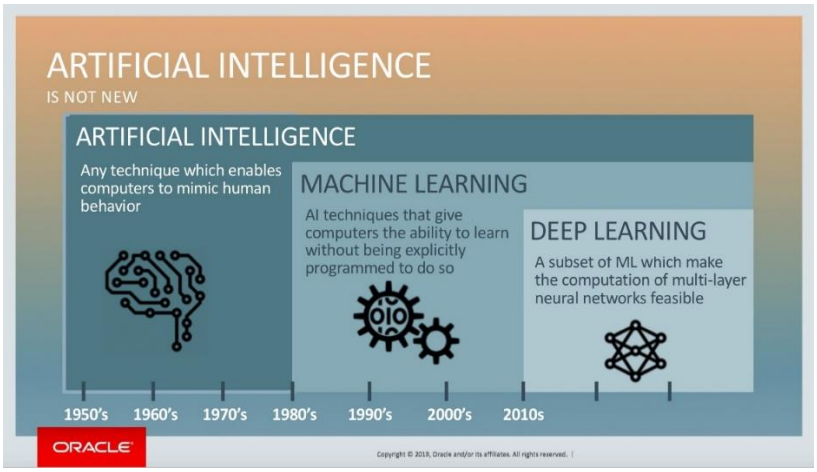

Figure 2.1 The time line of AI, ML and DL (Jeffcock, 2018)

For bringing the ability to think to the objects around us, firstly we had to have an idea of how to work neural networks in the human brain. In 1943, Walter Pitts and Warren McCulloch took the first step to fill this gap, with their "A Logical Calculus of Ideas Immanent in Nervous Activity" titled work. With this work, logician Walter Pitts and neuroscientist McCulloch created the first mathematical model of the neural network (McCulloch and Pitts, 1943).

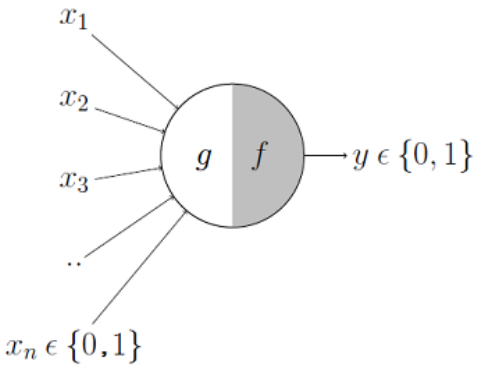

Figure 2.2 First mathematical model of neural network (Chandra, 2018)

In 1950 English mathematician Alan Turing mentioned the famous test. He called imitation game which tried to understand whether machines can think. In this test, which is referred to as the Turing Test, a machine must realize a written conversation

with any human. When the five-minute written conversation was over, if the person who was realising the conversation was convinced that the other thing is a human, the machine would passed the test. Thus, it would be concluded that the machine could think (Turing, 1950).

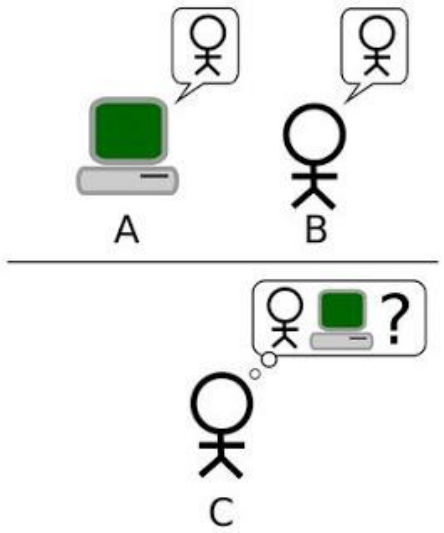

Figure 2.3 Illustration of turing test (Turing testi, 2021)

Arthur Samuel who used the machine learning term first time, was a advanced researcher of the artificial intelligence area. Till late 1960s, he did the good studies for making machines learn from their experiences (McCarthy and Feigenbaum, 1990). Samuel analyzed annotated games of checkers and created programs that could play checkers. Samuel was able to distinguish good moves from bad moves thanks to the algorithm he created. In addition, this algorithm had a self-training structure by controlling the past moves (McCarthy and Feigenbaum, 1990).

In the year 1957, Frank Rosenblatt's "The Perceptron: A Perceiving and Recognizing Automaton" titled article laid the foundations of deep neural networks. Maybe it wasn't a software approach but his work was a good hardware solution for this field. In this article, Rosenblatt mentions that he would create an electromechanical system that would learn to recognize similarities or identities between optical, electrical or sound information patterns in a way that is close to the perceptual processes of biological brain (Rosenblatt, 1957).

In 1959, David H. Hubel and Torsten Wiesel published "Receptive Fields Of Single Neurones In The Cat's Striate Cortex" as a biological development. In this work, they explained the concepts of simple cells and complex cells. This study has been a study inspired by researchers in the following years (Hubel and Wiesel , 1959).

When the calendars show the year 1960, aerospace and ocean engineer Prof. Dr. Henry Kelley published his article titled “Gradient Theory of Optimal Flight Paths”. Opinions about the theory of control were introduced in his work. These ideas laid the foundations of backpropagation algorithm that will be used frequently in the following years (Kelley, 1960).

In 1965, the first deep learning network, which until then couldn't go beyond of an idea, was created. Alexey Ivakhnenko developed the Group Method of Data Handling method. In this learning algorithm, multi-layer feed forward perceptrons that use statistical methods are used to find and highlight the best qualities in each layer (Ivakhnenko and Lapa, 1966).

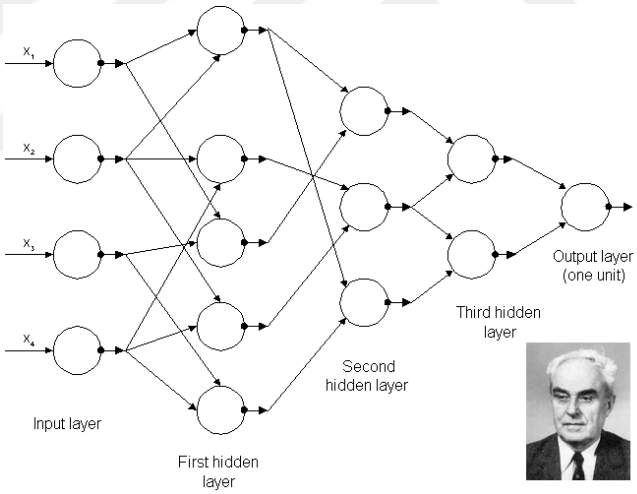

Figure 2.4 The architecture of the first known deep network (Dettmers, 2015)

In 1980, Kunihiko Fukushima published an article titled “A Self-organizing Neural Network Model For A Mechanism of Pattern Recognition Unaffected by Shift in Position”. In this article, a self-educating network with unsupervised learning has been developed. While creating this network, Fukushima was influenced by the

visual systems in animals. In addition, this network included the pooling layer which found in modern networks frequently (Fukushima , 1980).

The Hopfield network, an associative network model, was proposed in 1982 by John Hopfield in the article titled "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". This network has one layer unlike other artificial neural networks. The input and output layers was the same in his network (Hopfield, 1982).

In 1986, David Rumelhart, Geoffrey Hinton and Ronald Williams presented a study about backpropagation with an article titled "Learning Representations by Back-propagating Errors". They explained how far backpropagation algorithm can move forward networks that want to perform missions such as word prediction and shape recognition (Rumelhart et al., 1986).

In 1989, Yan Lecun et al published an article titled “Backpropagation Applied to Handwritten Zip Code Recognition” (Lecun et al.,1989). Successful results have been achieved with this network, which is a accomplished application that can read text on mailboxes. On the other hand, this network was the first network that are applied back propagation algorithm. However, the 3-day training period of the network has revealed the fact that it is not very convenient in practice (Şeker et al., 2017).

Again in the 1989 year, Christopher Watkins published his doctoral dissertation titled "Learning From Delayed Rewards". In his thesis, he introduced the concept of Q-learning, which increases the practicality of reinforcement learning (Watkins, 1989).

In 1995, Corinna Cortes and Vladimir Vapnik published their article titled “Support-Vector Networks”. SVM (Support Vector Machine), which uses manual features and works problem-specific, has entered our lives with this work. Artificial neural networks, which have high computational costs despite their advantages, have been shelved for a while. From the 1990's until the 2000's, SVM was preferred more (Cortes and Vapnik, 1995) (Şeker et al., 2017).

Also in 1995, Geoffrey Hinton, Peter Dayan, Brendan Frey and Radford Neal published the article titled “The Wake-sleep Algorithm for Unsupervised Neural

Networks". It proved that a network with many hidden layers can be trained if the length of the training time is ignored (Hinton et al., 1995)(Şeker et al., 2017).

In the year 1997, German computer scientists Jürgen Schmidhuber and Sepp Hochreiter introduced the concept of Long Short Term Memory (LSTM), which is a recurrent neural network structure. Schmidhuber and Hochreiter discussed the details of this method in their work titled "Long Short-Term Memory" (Hochreiter and Schmidhuber, 1997).

Yann Lecun, who has previously achieved impressive success in the field of deep learning, published his article titled "Gradient-Based Learning Applied to Document Recognition" in 1998. In his work Lecun showed that more accurate results can be obtained by combining the stochastic gradient descent algorithm and the backpropagation algorithm (Lecunn et al., 1998).

In the context of Artificial Neural Networks, deep learning statement was first introduced in 2000 with Igor Aizenberg, Naum Aizenberg and Joos Vandewalle' s "Multiple-Valued Threshold Logic and Multi-Valued Neurons" titled work (Şeker et al., 2017).

In 2007, Geoffrey Hinton, with his work titled "Learning Multiple Layers of Representation", showed that a multi-layer feed forward neural network can be trained efficiently in every iteration. In addition, this work presented that the efficiency would increase more as a result of the adjustments made on backpropagation algorithm (Hinton, 2007)(Şeker et al., 2017).

In 2009 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei announced that the ImageNet dataset were presented with the work titled "ImageNet: A large-scale hierarchial image database" published in IEEE Conference on Computer Vision and Pattern Recognition. At the time of its publication, it was contained a total of 3.2 million images and 5247 synsets (Deng et al., 2009).

In 2011 and 2012, Dan Ciresan et al. presented their work titled "Convolutional Neural Network Committees for Handwritten Character Classification" (Dan et al., 2011), "Multi-column deep neural network for traffic sign classification" (Ciresan et al., 2012) and "Deep Neural Networks Segment Neuronal Membranes in Electron

Microscopy Images" (Ciresan et al., 2012). They achieved significant success in competitions with their studies about character recognition and traffic signs.

In 2012, the Alexnet deep learning architecture, put forward by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton, achieved great results in the ImageNet competition. Using the dropout layer, which is an ideal method to reduce overfitting, has played an active role in this fascinating success (Krizhevsky et al., 2012).

In 2014, with GoogleNet deep learning architecture, Christian Szegedy and his friends were the winners of ImageNet (Szegedy et al., 2015). In 2015 Kaiming He and his team won the ILSVRC with ResNet deep learning architecture (He et al., 2016).

In the light of these developments, big technology companies noticed the future in the field of deep learning and they started to invest in this field. Deep learning progressed more rapidly with the investments made by leading companies, especially between 2012 and 2017 (Şeker et al., 2017).

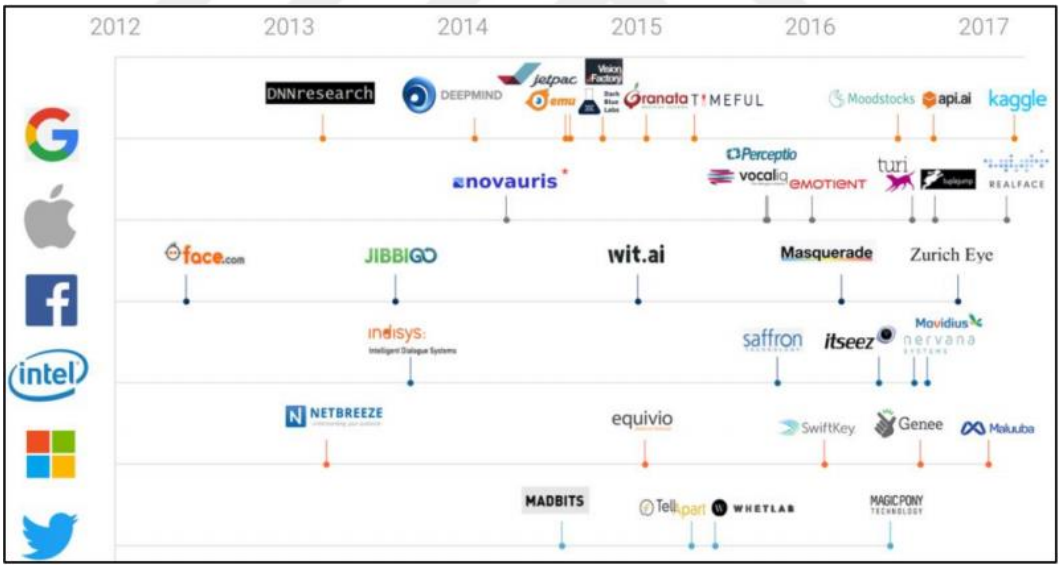

Figure 2.5 Purchase of AI startup companies by years (Şeker et al., 2017)

2.2 History of Convolutional Neural Network

Machine learning, whose foundations were laid in the 1950s and seen as a subset of artificial intelligence, achieved revolutionary success in many areas in the 2000s. What creates the deep learning method is neural networks, which are considered as a

sub-branch of machine learning. Deep learning, which has been developed mostly since 2006, has achieved great success in almost every field where it is applied. In addition, feature extraction and pattern classification processes are carried out with deep learning consisting of several hidden layers between the input and output layers (Schmidhuber, 2015)(LeCun et al., 2015). In some sources, deep learning is considered as a universal learning method that can solve many problems in different fields of study. Because, unlike SVM deep learning is not mission specific (Bengio, 2009).

CNN is a deep learning architecture that consists of one or more convolutional layers, pooling (downsampling) layer, which is usually used after the convolutional layer, and fully connected layers like a standard neural network (LeCun et al., 2015).

Inspired by David Hubel and Torsten Wiesel's study of simple cell and complex cell separation (Hubel & Torsten, 2009), Kunihiko Fukushima presented his study titled "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". This network structure was proposed in 1988 (Fukushima , 1980).

In 1988, the architecture named Lenet was proposed by Yann Lecun. This was the first working CNN architecture. Improvements on this architecture continued until 1998. Successful results were obtained in consequence of the operations performed with the MNIST data set consisting of handwritten numbers. However, it has not reached widespread use due to the long process time and insufficient equipment (Alom, et al., 2018).

CNN has been used in many different fields such as image processing, sound processing, natural language processing and biomedical research. CNN technique has achieved great success by proving itself especially in the field of image processing. In 2012 Ciresan et al. reduced the error rate to 2% by using the CNN technique and the MNIST data set in their studies (Ciresan et al., 2012).

In 2012, Alex Krizhevesky and his friends were the winners of the ILSVRC, which is considered to be quite challenging. After Lenet, which is first successful network architecture, AlexNet reduced the error rate very well. This great success enabled deep learning to rise from the ashes (Krizhevsky et al., 2012).

In the following years, ZFNet (Zeiler and Fergus, 2013), VGGNet (Liu and Deng, 2015) GoogleNet (Szegedy et al., 2015) and ResNet (He et al., 2016) achieved significant degrees in ILSVRC using the CNN technique.

In addition, important successes have been acquired in the field of game with the CNN technique. A pre-trained multi-layer CNN model played with a GNU Go algorithm beat the opponent with a great result of over 95% in the struggles (Gao et al., 2018).

AlphaGo was developed by Google DeepMind using CNN technique. AlphaGo defeated Fan Hui, who is a professional Go player, in 2016. Even, he was the champion of the world (Lee et al., 2016).

With all these developments, the CNN technique has gained an increasing popularity in the 2010s.

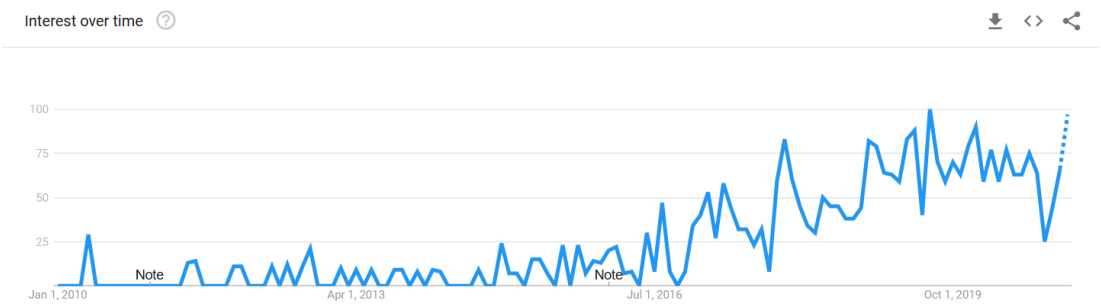

Figure 2.6 Interest over time to CNN (Google, 2021)

2.3 Related Works

In 2012, as mentioned earlier, the AlexNet architecture has had tremendous success in the ILSVRC, which was widely accepted. After this success, the CNN technique has become quite popular. With the developing technology and the increase in the number of data that can be processed, deep learning architectures with more layers have emerged. Teams that achieved significant degrees in ILSVRC after 2012 also used the CNN technique. In this study, hyperparameter optimization was performed by using CNN technique with GTSD. In this study, the effects of hyperparameters on the performance was examined through the design parameters created.

In the study published by Ayla Gülcü ve Zeki Kuş titled "A Survey of Hyper-parameter Optimization Methods in Convolutional Neural Networks"; Genetic Algorithms, Particle Swarm Optimization, Differential Evolution and Bayes Optimization meta-heuristic optimization methods were examined on several data sets such as CIFAR, EMNIST, MNIST, STL, UCL. By revealing the pros and cons of these methods, the points to be considered in the selection of hyperparameters are underlined. Gülcü and Kuş stated that it is not possible to make a generalization as choosing the best hyperparameter set. On the other hand, they said that hyperparameter optimization may vary according to the problem and data set. However, they showed that the hyperparameters that affect the result most are number of kernels, size of those kernels at each layer, dropout, number of layers, batch size and learning rate (Gülcü and Kuş, 2019).

Erik Bochinski, Tobias Senst and Thomas Sikora published their work titled "Hyper-parameter Optimization For Convolutional Neural Network Committees Based on Evolutionary Algorithms" at the 2017 IEEE International Conference on Image Processing. Bochinski and others stated that CNN is an outstanding method because of its superior performance, but finding the best performing network structure is quite important. The team, which carried out the experimental part of the study with the MNIST data set, presented an evolutionary algorithm-based study that can automatically optimize the CNN's network structure with hyperparameters (Bochinski et al., 2017).

Ferhat Kurt presented his study titled "Investigation of The Effect of Hyperparameters On Convolutional Neural Networks", which he prepared as his master's thesis in 2018. In his study, Kurt used the dataset used in ILSVRC 2012 and he made changes on the hyperparameters. After that, he trained the algorithms he obtained as a result of changes on hyperparameters. Kurt compared the data acquired in consequence of the training with graphs. He noticed that preprocessing the data set, choosing the learning rate according to the optimizer, using batch normalization and dropout process are increased the model performance (Kurt, 2018).

In 2016, Xiong Changzhen, Wang Cong, Ma Weixin and Shan Yanmei presented the study titled "A Traffic Sign Detection Algorithm Based on Deep Convolutional Neural Network" at the IEEE International Conference on Signal and Image Processing (ICSIP). They stated that the detection of traffic signs undertakes an

important task by increasing the safety in traffic. Changzhen et al. conducted the experimental part of the study with the Chinese Traffic Sign Dataset. The created model was tested with 33 different videos with a width of 640x480 and it make over 99% accurate predictions (Changzhen et al., 2016).

In 2020, Sheikh Shanawaz Mostafa, Fábio Mendonça, Antonio G. Ravelo-Garcia, Gabriel Gabriel Juliá-Serdá, Fernando Morgado-Dias; presented the study titled "Multi-Objective Hyperparameter Optimization of Convolutional Neural Network for Obstructive Sleep Apnea Detection". Mostafa et al. used the CNN technique to detect obstructive sleep apnea, which causes breathing interrupted during sleep. They have been stated that shallow neural networks are inadequate due to the need for feature creation and the selection of the more relevant features. Also they showed correct results can be obtained with appropriate hyperparameter selections. In this study, three different input sizes and different data set were tested. As a result of the tests, the best model reached an average accuracy, sensitivity, and specificity of 94%, 92%, and 96%, respectively (Mostafa et al., 2020).

Viktoria Plemakova from Tartu University completed his master thesis titled "Vehicle Detection Based on Convolutional Neural Networks" in 2018. In this study, Plemakova presented a model that detects and classifies the images of vehicles at different angles on the data set obtained by combining the different data sets which inclusive and noninclusive car pictures. He used the CNN as main technique of his work. He also observed the effect on the result by using Fast Fourier transform during data preprocessing (Plemakova, 2018).

CHAPTER 3

METHODOLOGY AND TOOLS

In the third chapter we are in, some technical issues are explained in order to increase the understandability of this thesis. Computer vision, deep learning, artificial neural networks and convolutional neural networks subjects are mentioned and a suitable ground was prepared for the next chapter. In addition, tools used in this study are clarified in this section.

3.1 Computer Vision

Computer vision is defined as a field of study that searchs to improve some techniques to help computers see and perceive the content of digital images like photographs and videos (Brownlee, 2019). Therefore, computer vision is concerned the extraction of meaningful information from the digital sources like images or videos. Computer vision's applications areas include image classification, image retrieval, traffic automation, visual detection, 3D scene reconstruction from 2D images, augmented reality and machine vision (Szeliski, 2010). Machine learning is a obligatory component of numerous computer vision algorithms in a lot of study. Created models, which is in this field, can be defined as a compound of machine learning and image processing. Models that can process large amounts of information in visual data in real time are very important for many applications (Sebe et al., 2005)(Huang, 1996).

Lawrence Roberts, who discussed the possibilities of extracting 3D geometrical information from 2D perspective views of blocks in his Ph.D. thesis work (Roberts, 1963) at MIT, is admitted that founder of computer vision according to many researcher. Later, researchers become aware of that it was required to handle data from the real world. For this reason, low-level vision tasks such as edge detection and segmentation were needed (Huang, 1996).

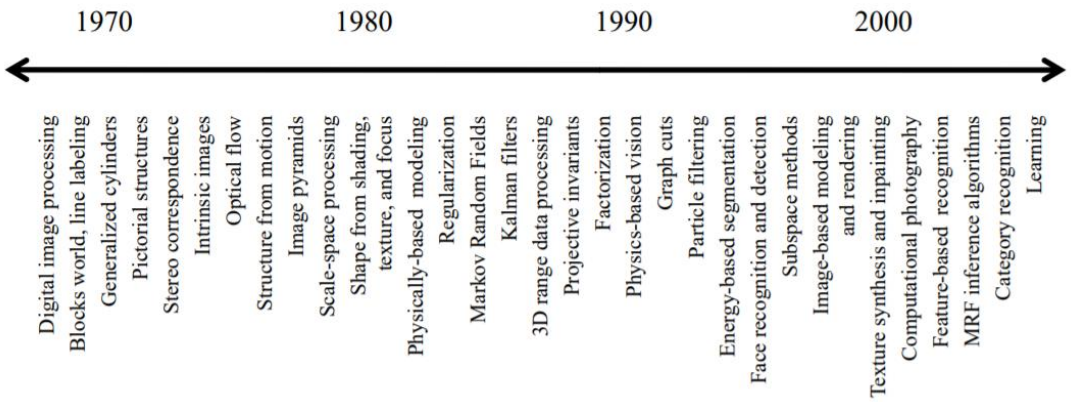

Figure 3.1 Timeline of the most active topics of research in CV (Szeliski, 2010)

3.1.1 Object Detection

Object detection is already a significant challenging computer vision task. Object detection is a serious part in a lot of applications like image search, image auto-annotation and scene understanding; nevertheless from many perspectives it is an sophisticated problem. It is still challenging, especially due to the complexity of object classes and images (Wang et al., 2007).

Object detection causes some kind of chicken-egg paradox in some cases. When we want to detect the object in any image, we should have an idea of where the object might be or how the picture is segmented. Because we need to know the position of the object in the picture to recognize its shape. Similarly, to understand its position, we must know the shape of the object (Walther et al., 2002).

Generally bounding box is used, while any object is being detecting in an image. Bounding box, which is usually selected as rectangle shape, includes location and size information of object that we want to detect (Girshick et al., 2014).

A natural property of objects around us is that they only exist as meaningful presence in specific ranges of scale. We can give a tree as an example. Maybe it won't reach the size of kilometers or nanometers. But different types of trees, even trees of the same species, can be of very different sizes. For reasons like this, in the 2000s, feature descriptors were benefitted for solving object detection problems. Methods

such as Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF) and Histogram of Oriented Gradients (HOG) have been used to provide solutions to image rotation, affine transformations, intensity, and viewpoint change features, just like scale (Lindeberg, 1994) (Karami et al., 2015)(Stenroos, 2017). In the 2010s, CNN became a more preferred method (Girshick et al., 2014).

3.2 Deep Learning

In ancient Greek myths, intelligent objects that act on command are mentioned. This idea has been one of the greatest desires of mankind ever since. When programmable computers were invented, people wondered whether they could think or not (Menabrea, 1842)(Mandelbaum, 2004).

Especially recent years, artificial intelligence has been a successful field in many practical applications. Most of the early successes of artificial intelligence occurred generally in more shallow fields. Computers that were expected to succeed did not need to have extensive knowledge about the world. IBM's Deep Blue chess-playing system can be an example. Deep Blue defeated world champion Garry Kasparov in 1997. However, as is known, chess is not a very large area and it has formal rules. Chess consisted of 32 pieces, which are moved with limitted movement, and 64 squares (Tomayko, 2003).

Hard-coded systems do not have the ability to obtain their own information. However, systems that can learn their own knowledge by extracting information from raw data are more valuable. The area where this skill is demonstrated is machine learning. The performance of simple machine learning algorithms is related with the representation of the data which they are given. Surely, it can be overwhelming to extract high-level intangible features from raw data. However, deep learning makes possible the computer to construct complex concepts from simpler concepts. Figure 3.2 shows a flow chart of a deep learning model. It is very difficult for a computer to understand what the object in the picture is. Deep learning method, which performs a different function in each layer, can be used to solve this problem. Since there are features that can be observed in the first layer, this layer is named as the visible layer. After this layer, there are one or more hidden layers which higher level features are extracted. The reason of these layers are called hidden is that the values were not given before. We can obtain to features which we can get

information about the object in the picture through these layers. In the first layer, edges can be detected by comparing the brightness of neighboring pixels. The second layer can detect corners and contours, which can be treated as the collection of the edges found in the first layer. In the third layer, all parts of the object can be detected by combining corners and contour information. Then what is the object in the image is detected in the output layer (Goodfellow et al., 2016).

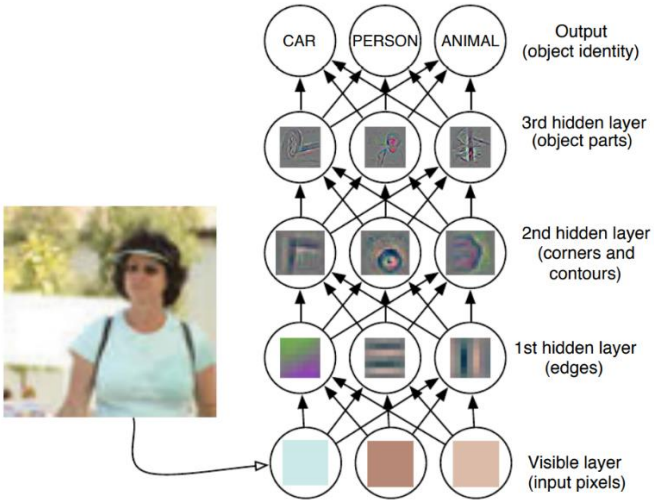

Figure 3.2 Illustration of the deep learning model (Goodfellow et al., 2016)

3.3 Artificial Neural Network

As the name suggests, Artificial Neural Networks (ANNs) are computational networks that emulate the central nervous system of people and animals (Wang S. C., 2003). Today, ANNs are widely used in many areas such as manufacturing, security, health, transportation, energy, marketing, insurance, finance, education, science, agriculture. The use of ANN in these areas is a search for a solution beyond traditional computational capabilities and mathematics (Abiodun et al., 2018).

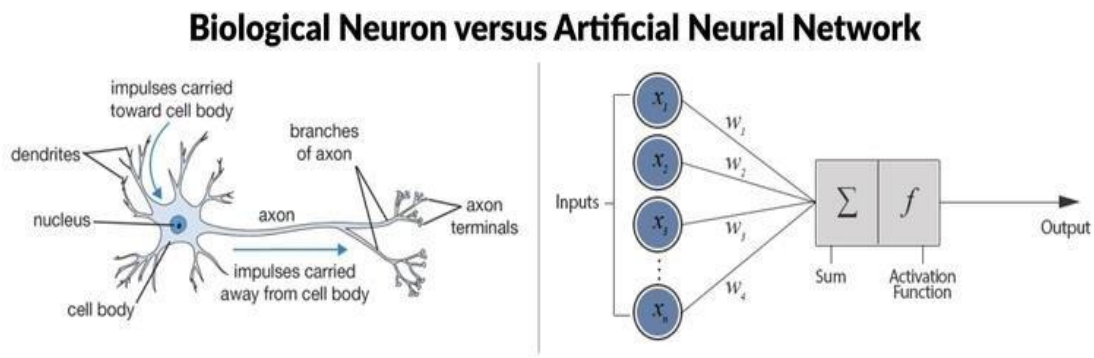

Figure 3.3 Biological neuron and artificial neural network (Willems, 2019)

ANNs are parallel computing systems consisting of an extremely large number of simple processors with a large number of interconnections. ANN, which is used for engineering purposes such as recognition and forecasting, is inspired by the principles thought to be used in neural networks of living things. An ANN model consists of these parts; input layer of nodes, one or two hidden layers of nodes and finally output layer. Inputs similar to synapses in biological structure are multiplied by weights. Weights contain numerical numbers and represent the flow of information. The values obtained after multiplication are summed and then the result is obtained with the help of a mathematical function called the activation function. By adjusting the weights correctly, we can get the result we want for certain inputs. This process of adjustment is called learning or training (Gupta, 2013)(Wang S. C., 2003).

3.4 Convolutional Neural Networks

Deep Learning or deep neural network terms refer to multi-layer artificial neural networks. Deep learning techniques with their gradually deepening structures have achieved more successful results compared to traditional structures. One of these successful techniques is CNN, which has proven itself in many areas where it is applied such as pattern recognition, image classification, natural language processing (Albawi et al., 2017).

3.4.1 What is Convolutional Neural Network

For the basic working principle of CNN, it is completely inspired by the processes of seeing and perceiving objects around living things. As a living form, people perceive everything around them with all their colors and details. However, until this process occurs, many processes take place in the background. There are photoreceptors in our visual system in the area called the retina. Some of these receptors are sensitive to wavelength, which makes them color sensitive. Some receptors are also sensitive to light intensity. These receptors, which are linked together, are positioned that the photoreceptors that react to light are surrounded by the photoreceptors that react to the dark. All of these receptors bind to a ganglion cell. These ganglion cells lined up in a straight line are connected to a cell called a simple cell. With these structures coming together and working in a coordinated manner, we determine the concrete properties of the objects around us. CNN carries out this process just like the biological background. First, it extracts low-level features and learns them. It then extracts higher-level features. Thus, perhaps without even noticing it, we realize the process of perception by finding unique features that we dedicate to the object we want to perceive and enable us to distinguish that object from others. For example, when we look at a picture of an airplane, we can say that what we are looking at is the plane, using distinctive features such as two wings or windows (Barmettler, 2019)(Rubik's code, 2018).

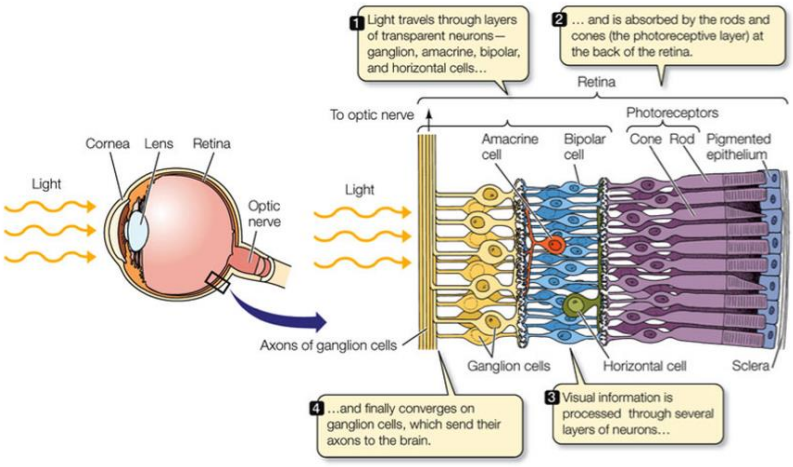

Figure 3.4 The illustration of basic vision system of animals (Brutus, 2018)

It can be considered that CNN is similar to ANN in that it consists of self-optimizing neurons. The main point that CNN differs from ANN is that it first determines the features of the object in the image. This distinction not only leads to better results but also provides advantages for calculation cost. MNIST data set is frequently preferred, especially by beginners. If it is desired to operate using this data set and traditional ANN, even the first hidden layer will contain 784 weights. MNIST data set consists of images in gray scale and 28x28 size. However, if an image with 3 channels and size of 227x227 is used as an input, this number will increase up to 154587 (O'Shea & Nash , 2015). In CNN, the size of the kernels (filters) used for feature extraction plays a decisive role in the calculation cost.

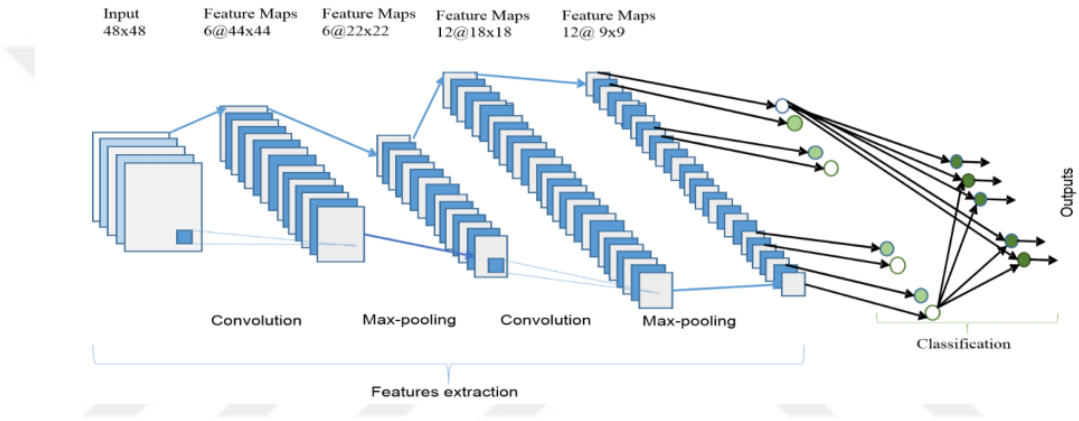

Figure 3.5 Illustration of any CNN architecture (Alom et al., 2018)

3.4.2 Layers of Convolutional Neural Network

Generally, CNN has three main layers. These layers are convolutional layer which is used for feature extraction, pooling layer that is used for downsampling and fully-connected layer which implements tasks also found in traditional ANN (O'Shea and Nash , 2015).

3.4.2.1 Convolutional Layer

CNN uses convolution operation instead of general matrix multiplication in at least one layer. The reason why it is called convolutional neural network is the realization of convolution, which is a mathematical process (Goodfellow et al., 2016).

A convolution layer is a main component of the CNN technique that performs feature extraction. In this layer, matrices called filters or kernels are used. Element-wise multiplication operation is taken place between input tensor and kernel. Feature extraction is performed with the help of these filters and feature map is obtained as a result of this process. Filters perform the convolution process, just like a flashlight, passing over all the pixels in the input picture. Different useable features can be extracted by using different filters. Then, as in the basis of deep learning, low-level features are combined to obtain high-level features in the next layers. As a result of this process with filters, only important information is obtained and unnecessary information is eliminated. Because when the feature map is obtained, the matrix, which we have, contains only important information (Alom et al., 2018)(Yamashita et al., 2018) (LeCun et al., 2015).

The distance between two consecutive filter positions is called stride. It is usually chosen as one. However, in some cases, bigger value can be chosen. When it is chosen bigger value, the goal is to perform more downsampling. For instance when we set the stride hyperparameter as one, we would have a seriously overlapped receptive field producing quite big activations. However, if we were set the stride to a bigger number will reduce the amount of overlapping and produce an output which has lower spatial dimensions (Yamashita et al., 2018).

In this layer, when focusing only on pixels that contain important information, data may be lost. The same spatial dimension at the input and output may also be desired. In this case, zero padding is used. For example, suppose that a 9x9 size image is used as input data and a 3x3 matrix is used as a filter. The size of the feature map to be obtained after the convolution process will be 7x7. However, if zero padding is used, the spatial sizes of input and output matrices will be the same (Albawi et al., 2017).

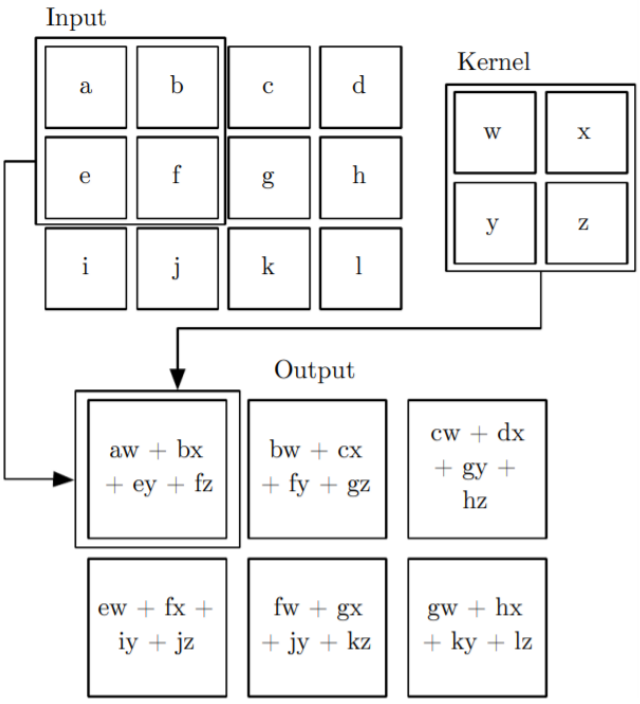

Figure 3.6 The Illustration of the convolution operation (Goodfellow et al., 2016)

$$O = \frac{I+2p-f}{s} + 1 \qquad (3.1)$$

In the above formula, the output is shown with “O”, the input picture with “I”, the filter size with “f”, the padding “p” and the stride with “s”. This formula gives the spatial dimension of the output image to be obtained as a result of the operation with the selected hyperparameters.

3.4.2.2 Pooling Layer

Another fundamental CNN layer is the pooling layer. The main purpose of using this layer is to reduce complexity in subsequent layers and to highlight pixels that contain more important information. Although it can be applied in different methods such as max pooling and average pooling, max pooling is more commonly used. When using max pooling, it splits the image into small pieces and takes the largest value in each piece. In average pooling, it takes the average of all the values in each piece. There are no learnable parameters in the pooling layer. However, parameters like filter size

and stride are similarly hyperparameters as in convolutional layer (O'Shea and Nash , 2015)(Yamashita et al., 2018).

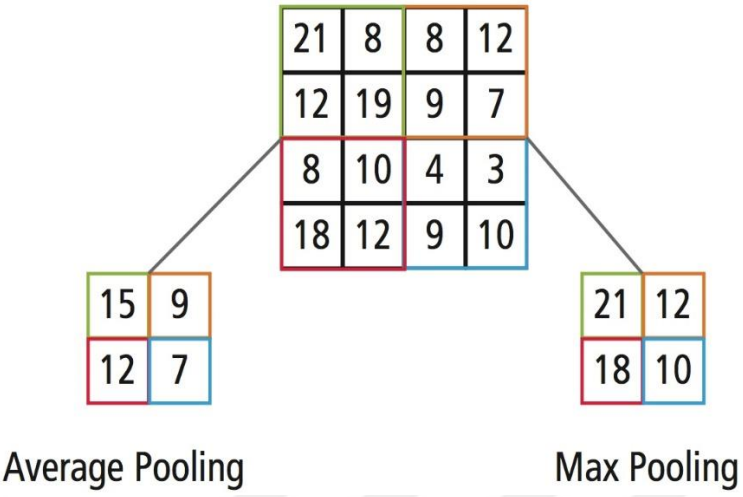

Figure 3.7 Illustration of pooling layer method types (Mualla & Alkheir, 2018)

3.4.2.3 Fully-Connected Layer

The feature map, which is obtained after the last convolution or pooling layer, is flattened. One-dimensional numeric array is obtained after flattening. This layer, which is found in one or more numbers in architectures, is also called dense layer. Each node in the fully-connected layer is directly connected to each node in the previous and next layers. This situation is shown in Figure 3.6. In this respect, the fully connected layer is similar to the standard neural network (Dumoulin and Visin, 2016) (Kwak, 2016).

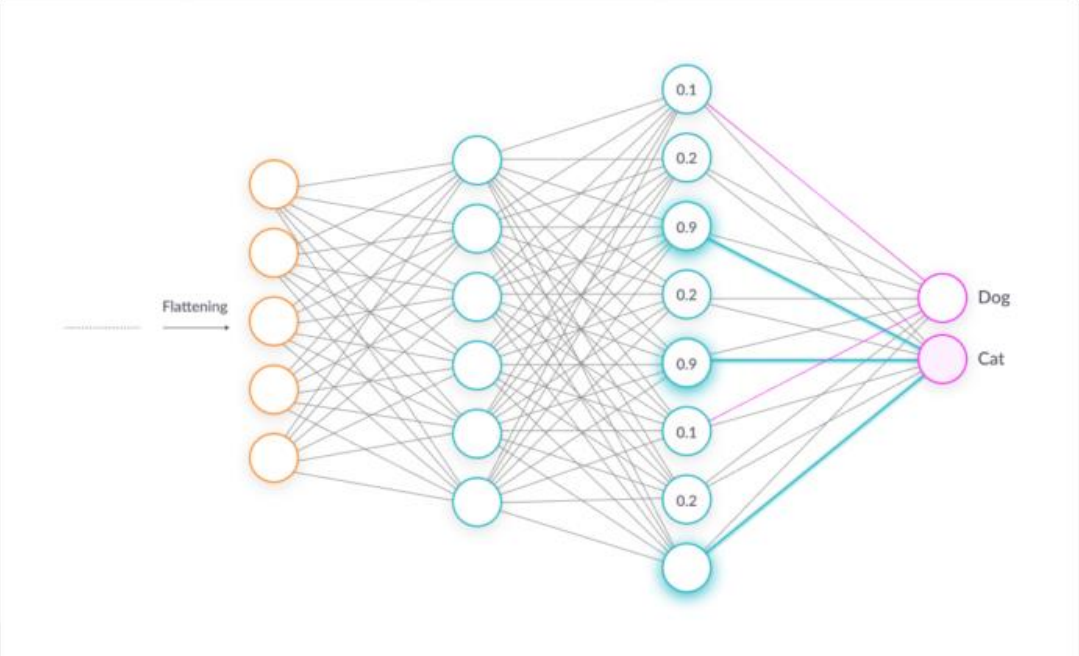

Figure 3.8 Illustration of fully connected layer (Chen, 2020)

The fully connected layer has the same number of output nodes as the number of classes. The activation functions in this layer are slightly different from the other layers. Acquired results with the selected activation function such as sigmoid, tanh or softmax, the value of all possible outcomes in the class must be between 0 and 1, and the sum of all these probabilities must also be one (Yamashita et al., 2018).

3.4.3 Activation Function

In artificial neural networks, input values and corresponding weights are multiplied. Then these multiplication results are summed and the activation function is applied to the obtained value. The correct selection of this function plays a critical role in the performance of the model. The activation function generates an output signal from an input signal and thus enables data to be transmitted to the next layer (Sharma et al., 2020).

$$y = \alpha(w_1 x_1 + w_2 x_2 + .. + w_n x_n + b) \quad (3.2)$$

In the above formula, symbols are represented as follows; "y" output, "α" the activation function to be applied, "w" weight, "x" input and "b" bias.

There are many types of activation function. These activation functions can be listed as following;

- Binary Step Function,
- Linear,
- Sigmoid (Logistic),
- Tanh,
- ReLU,
- Leaky ReLU,
- Parametrized ReLU,
- Exponential Linear Unit,
- Swish and
- SoftMax. (Nwankpa et al., 2020)

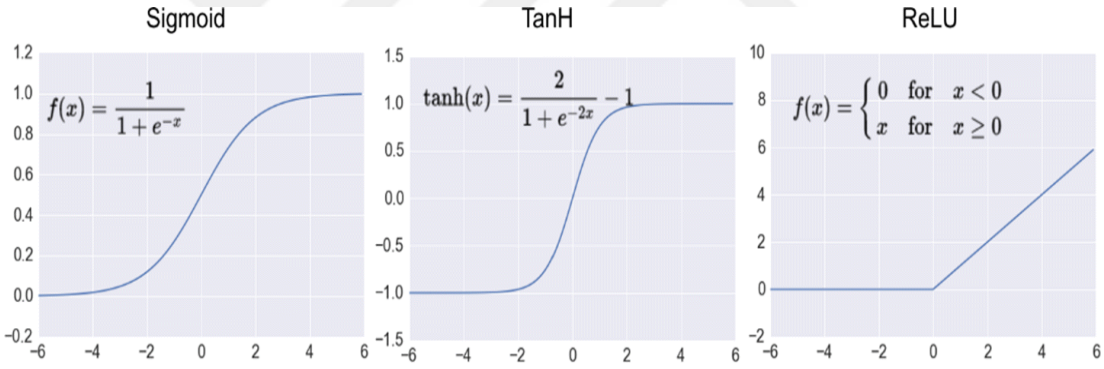

Figure 3.9 Plotting and formulas of sigmoid, tanH and ReLU activation functions (Maalej and Kherallah, 2020)

Table 3.1 Activation Functions Used by Some Popular DL Architectures

Architecture	Hidden Layers	Output Layer
AlexNet	ReLU	Softmax
Nin	No Activation	Softmax
ZFNet	ReLU	Softmax
VGGNet	ReLU	Softmax
SegNet	ReLU	Softmax
GoogleNet	ReLU	Softmax
ResNet	ReLU	Softmax
SeNet	ReLU	Softmax

3.4.4 Back-propagation Algorithm

When using a feed forward neural network, an output is obtained from the input information. As shown in Formula 3.2, the weights should be added after multiplying by the inputs and the bias value should be added to this sum. In the last step, output is obtained with the help of the activation function. If no feedback is implemented, the result may not be satisfactory. Back propagation algorithm (Rumelhart et al., 1986) comes into play at this point. Just like in the supervised algorithm, input and output samples are given to be calculated. Then the error is calculated with these data. What is stated as error here is the substraction between the result we have and the result we want to acquire. The network is then retrained by updating the parameters. The goal is to reduce the error until the system learns the training data (Gershenson, 2003).

We can consider backpropagation algorithm mathematically only as a gradient computing tool. It uses chain rule to perform this operation (Goodfellow et al., 2016).

3.5 Popular Convolutional Neural Network Architectures

When popular CNN architectures are examined, most architecture consists of convolution layers, pooling layers and finally fully-connected layers. In the last layer, output is obtained by using the softmax function (Alom et al., 2018).

LeNet architecture was introduced in the 1990s by Yann LeCun et al. At that time, it could not be implemented constantly due to the lack of sufficient technological infrastructure. In this architecture created using the backpropagation algorithm, there are 2 convolution layers, 2 pooling layers and 2 full-connected layers. (LeCun et al., 1998)

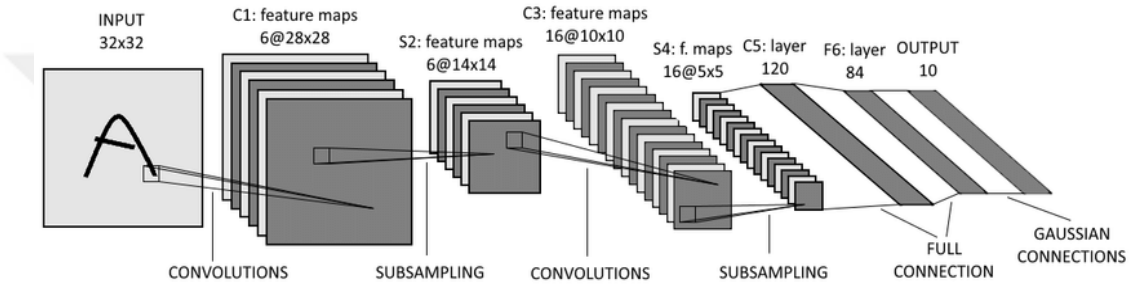

Figure 3.10 Illustration of LeNet-5 architecture (Ayumi et al., 2016)

AlexNet architecture (Krizhevsky et al., 2012) was introduced by Alex Krizhevsky and colleagues in 2012. This architecture, which runs on the ImageNet database containing 1.2 million images in 1000 different categories, was the winner of ILSVRC 2012. AlexNet architecture includes five convolutional layers, three sub-sampling layers and two full-connected layers. The use of the dropout layer also played an important role in this success (Shin et al., 2016).

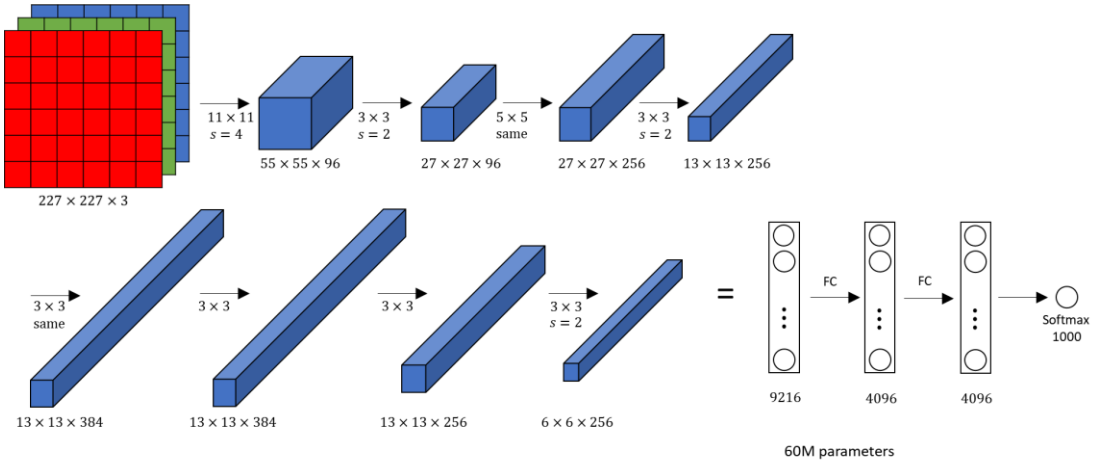

Figure 3.11 Illustration of AlexNet architecture (Data Hacker, 2018)

In 2013, Matthew Zeiler and Rob Fergue won the ILSVRC with ZFNet (Zeiler and Fergus, 2014) The network they have created is shown in Figure 3.12 and it is an improved version of AlexNet. In ZFNet, 7x7 filters are used instead of 11x11 filters. The number of weight parameters decreased significantly due to this change. Thus, an increase in accuracy was achieved (Alom et al., 2018).

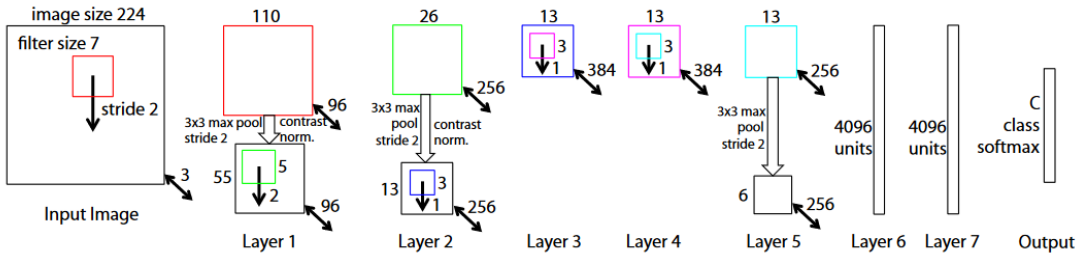

Figure 3.12 Illustration of ZFNet architecture (Zeiler and Fergus, 2014)

Visual Geometry Group launched VGGNet (Simonyan and Zisserman, 2014) in 2014. With this study, it was revealed that with a deeper network, more successful results can be obtained in CNN. While softmax activation function is used for classification in the last layer, ReLU function is used in hidden layers. In this architecture two convolutional layers, one max-pooling layer and several fully-

connected layers are used. VGGNet ranked second in ILSVRC in 2014 (Alom et al., 2018).

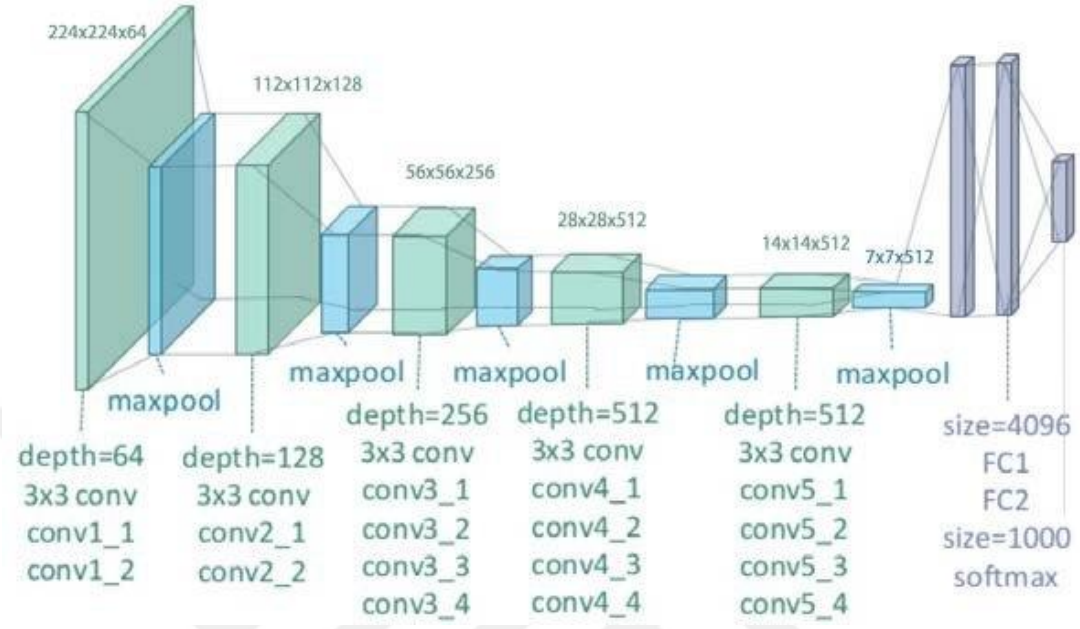

Figure 3.13 Illustration of VGG-19 architecture (Zheng et al., 2018)

In 2014, GoogLeNet (Szegedy et al., 2015) became the winner of ILSVRC with the 5.5% error rate. This network had a more complex and deeper structure than other models before it. In this architecture, the inception module is presented as different from the others. This module creates a new filter by combining filters of different sizes. Also in this architecture, there are two convolution layers, two pooling layers and nine inception layers (Shin et al., 2016).

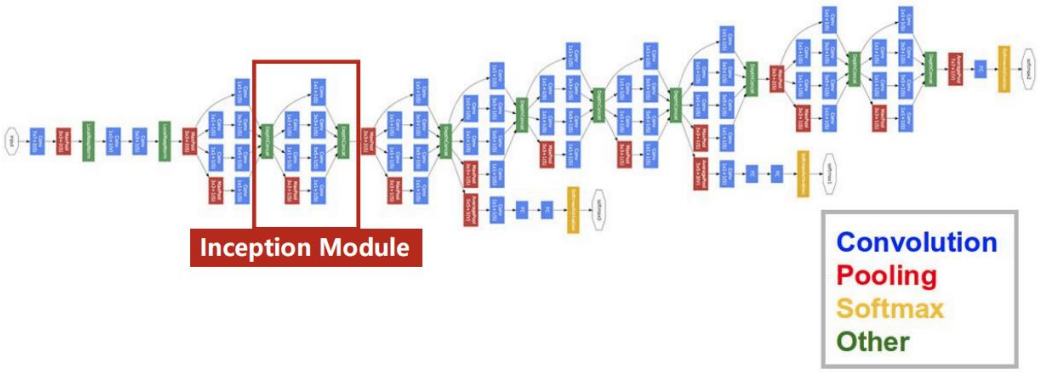

Figure 3.14 Illustration of GoogLeNet (Nongjiayuan, 2017)

Residual Network Architecture, called ResNet, won first place in ILSVRC in 2015. ResNet was developed by Kaiming He et al. This network includes a total of 25.5M weights and 3.9M MAC numbers (He et al., 2016).

Table 3.2 Computational parameters and MACs for some popular CNN architectures

Methods	**LeNet-5**	**AlexNet**	**VGG-16**	**GoogLeNet**	**ResNet**
Error Rate in ILSVRC	-	16.4	7.4	6.7	5.3
Input Size	28x28	227x227	224x224	224x224	224x224
Number of Conv. Layer	2	5	16	21	50
Filter Size	5	3,5,11	3	1,3,5,7	1,3,7
Number of Feature Maps	1.6	3-256	3-512	3-1024	3-1024
Stride	1	1,4	1	1,2	1,2
Number of Weights	26k	2.3M	14.7M	6M	23.5M
Number of MACs	1.9M	666M	15.3G	1.43G	3.86G
Number of FC Layers	2	3	3	1	1
Number of Weights	406k	58.6M	124M	1M	1M
Number of MACs	405k	58.6M	124M	1M	1M
Total Weights	431k	61M	138M	7M	25.5M
Total MACs	2.3M	724M	15.5G	1.43G	3.9G

3.6 Hyperparameters

To get better performance from any model, hyperparameters should be adjusted correctly. Parameter is a variable that is automatically set during the training process. However, the hyperparameter is a variable set by the user when creating the algorithm. In CNN architecture, there are many hyperparameters in each layer as shown in Table 3.3. There are no specific values in hyperparameter selection. These

are the variables that need to be adjusted according to the data set or the subject studied.

Learning rate can be defined as the coefficient that determines how much we will change the weights of our network. When the learning rate is chosen small, one can be more sure that local minimums are not missed. However, in this case, the training time will increase. When learning rate is chosen greater, it will not be possible for the model to train itself efficiently (Zulkifli, 2018).

Passing all data in the data set only once is called an epoch. There are usually more than one epoch number in the architectures created. Even though the system sees the all data in the data set, going over the same data set over and over again causes the model to be better trained. Instead of processing the entire data set at the same time, we can split it into pieces and process it. Each piece here is called batch size. Iteration is the number of batch sizes required to complete an epoch. For example, suppose we have 5000 image data in our data set. If we decide to have a batch size of 500, an epoch will be completed after 10 iterations (Sharma S. , 2017).

Table 3.3 Some parameters and hyperparameters in a CNN

	Parameters	**Hyperparameters**
Convolutional Layer	Kernels	Number of kernels, kernel size, stride, padding, activation function,
Pooling Layer	-	Pooling method, filter size, stride, padding
FC Layer	Weights	Activation function, number of weight
Others		Learning rate, epochs, loss function, batch size, optimizer, dataset splitting, input image size, dropout

3.7 Tools

Some tools were needed to complete this thesis work. A suitable data set, a software language, a deep learning library, a computer with sufficient technological infrastructure and other libraries were used as tools.

3.7.1 Data set

Although the foundation of deep learning was laid in the 1950s, it has become an indispensable technology after the 2010s. It was also used for commercial purposes in the 1990s, but at that time it was considered as method that can be used by experts. In this method, some skills were required to achieve good performance. In fact, this situation was somewhat justified this point of view. However, the increase in the available data has reduced the skill required to use this method. The introduction of the big data term into our lives has increased the machine learning and performance acquired from studies (Goodfellow et al., 2016).

For this study, a suitable data set research was conducted. While making a selection, attention has been paid to the fact that the data set to be used is comprehensive enough to make generalizations and that it does not increase the calculation time for each design parameter.

Recognizing traffic signs is a very important and challenging task for autonomous driving systems. These systems are expected to give accurate results even in misleading situations such as low light intensity or reverse angle. Citlalli Gamez Serna and Yassine Ruichek made a classification process using data sets belonging to Belgium, Croatia, France, Germany, Netherlands and Sweden with the study titled "Classification of Traffic Signs: The Europen Dataset" (Serna and Ruichek, 2018).

Table 3.4 Image numbers of some europan traffic sign dataset

Country	Dataset	Class Number	Training Images	Testing Images	Total Images
Belgium	KUL	62	4561	2528	7089
Croatia	MASTIF-2009	97	4568	1843	6411
Netherlands	RUG	3	58	17	75
Sweden	STS-Set2	19	2409	1033	3442
France	UTBM	10	203	68	271
Germany	GTSD	43	39209	12630	51839

Russian Traffic Sign Dataset contains over one hundred thousand pictures in total. Although there are such options like RTSD, GTSD has been preferred because it has sufficient amount of data.

Figure 3.15 Illustration of GTSD Image Classes (Rajesh et al., 2011)

3.7.2 Python

A software language is an artificial language which is designed by people. It used for communicating directives to a machine or a computer (Devero, 2015). Today, the software language learning process starts from the primary school level. Many electronic devices around us fulfill their duties with efficient software. It is necessary

to choose an appropriate programming language for the projects and studies to be carried out. For this, there are many software languages such as Python, C, C ++, C #, Arduino, Go Java, JavaScript, Assembly R and Matlab.

Figure 3.16 Ranking of software languages from 2016 to 2020 (Cass, 2020)

In this study, an appropriate programming language was needed while creating design parameters. It was necessary to choose one of the options such as Python, Matlab, Java, C / C ++ and R, which are frequently used in the field of artificial intelligence. Python, which is being high-level language, can work in harmony with many deep learning libraries. The Python software language was chosen because it is free, popular, easy to program and fast, as well as having a lot of resources.

3.7.3 Keras

One of the biggest problems when writing code is to avoid complexity. Libraries frequently used in software languages serve this purpose. Instead of rewriting some code, we can use the appropriate library. In addition, when used the library, it will be sufficient to dominate the interface instead of knowing all the details of related code.

(Wikipedia, 2021) In this study, many libraries, especially Keras, OpenCV, scikit-image, matplotlib, numpy, pandas, were used in the algorithm creation stage.

Keras, which means horn in Greek, is a TensorFlow based deep learning Application Programming Interface (API). Written in the Python programming language, this API is designed with a focus on fast learning and fast results. For this reason, it is easier to learn and apply. Keras is based on layers and models. Architecture can be easily created using the sequential model (Keras, 2021).

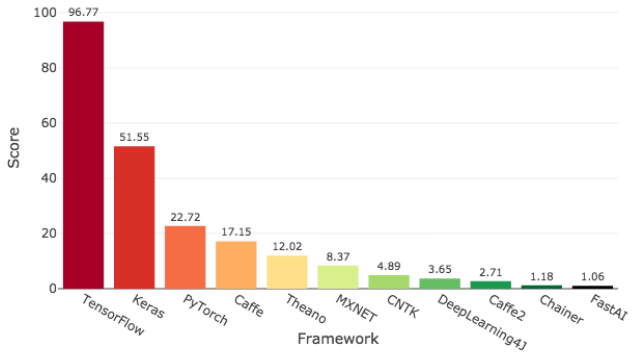

Figure 3.17 Deep learning framework power scores in 2018 (Hale, 2018)

3.7.4 Other Tools

Nitro AN515-54 model computer of Acer was used to obtain the experimental results of the design parameters. This device has system features such as 8 GB Ram, Intel Core i7-9750h processor with 2.60 GHz speed, 4 GB Nvidia GeForce GTX 1650 graphics card and 128 GB SSD storage. Also, Linux based Ubuntu 20.04 was used as the operating system.

In some design parameters, the aforementioned computer was unable to execute related code. For this, a more powerful device was searched and a server was used. The features of this device are as follows; 112 x Intel (R) Xeon (R) Gold 6238R CPU @ 2.20 GHz (2 Sockets), 755 GiB Ram and 94 GiB SDD storage. On the other hand, Ubuntu 18.04 operating system was used in this machine while executing design parameters.

CHAPTER 4

DESIGN PARAMETERS AND PERFORMANCE ANALYSIS

This section is where the fundamentals of this thesis are presented. Firstly, the methods that enable us to evaluate the performance of design parameters were mentioned. Then, it was explained how the created reference design parameter is decided. After that, the performance results of other design parameters obtained as a result of the changes made in the hyperparameters were shown in tables and graphics. Finally, analysis was made by comparing the acquired results.

4.1 Performance Measures Methods

We can interpret the performance results obtained for any model with the accuracy, precision, recall and f-score metric parameters. We know which label the data we will put into training has and which class it is in. Thus, according to the output given by the model, we can understand whether it has made the right decision or not. At this point, we use performance measurement methods with the help of different formulas using a confusion matrix.

Table 4.1 Confusion matrix

		Prediction Result	
		Negative	**Positive**
Actual Situation	**Negative**	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

TP and TN are situations in which the model predicts correctly. FP and FN are situations in which the model predicts incorrectly (Joshi, 2016).

$$Accuracy = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} \quad (4.1)$$

$$Precision = \frac{\text{TP}}{\text{TP+FP}} \quad (4.2)$$

$$Recall = \frac{\text{TP}}{\text{TP+FN}} \quad (4.3)$$

$$F - Score = 2 \text{ x } \frac{\text{Recall x Precision}}{\text{Recall+Precision}} \quad (4.4)$$

Precision, recall and f-score metric parameters have been removed as from Keras version 2.0. For this reason, these parameters are defined as functions and calculated manually.

4.2 Selection of Reference Design Parameter

In studies conducted in the field of deep learning, choosing the correct hyperparameter is quite important in terms of performance. Although there are no certain hyperparameter values that can be used for maximum performance, it is possible to generalize about the correct configuration adjustment.

In this thesis work, thirty-five different design parameters were applied on GTSD except Lenet-5 and AlexNet architectures. One of these design parameters has been chosen as the reference design parameter. This model, which is called Design Parameter (DP) 1, is compared with other design parameters and the effect of hyperparameters on the result was shown.

Popular CNN architectures were reviewed before creating the reference algorithm. The examination has started with LeNet-5 and AlexNet architectures. After the results obtained with these popular architectures, it was thought that the network to be created should give higher performance results than LeNet-5 and lower than

AlexNet. As shown in Figure 4.1, DP1 consists of four convolutional layers, two pooling layers and two fully-connected layers.

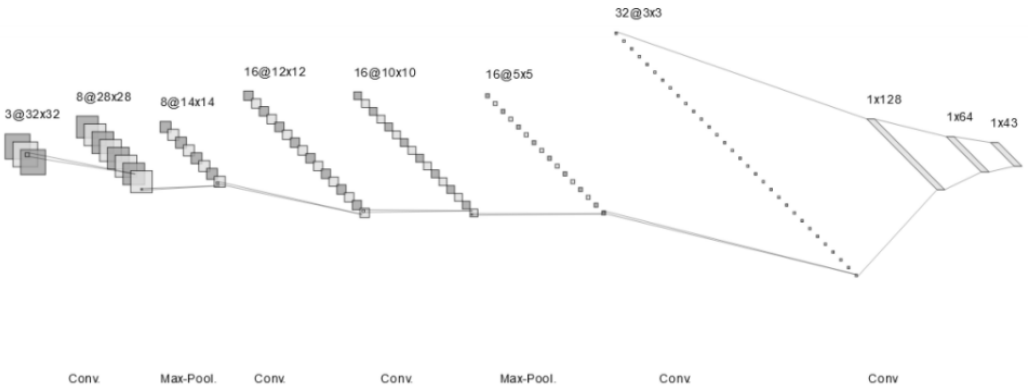

Figure 4.1 Illustration of reference design parameter architecture

Hyperparameter values of Lenet-5, AlexNet and design parameter 1 networks were shown in Table 4.2. In Table 4.3, information including the performance results of these architectures was given. The graphical output of these architectures was shown in Figure 4.2, Figure 4.3 and Figure 4.4.

Table 4.2 Hyperparameter configuration of Lenet-5, AlexNet and DP1

	LeNet-5	**AlexNet**	**DP1**
Learning Rate	0.01	0.01	0.01
Epoch	15	30	15
Batch Size	32	32	32
Filter Numbers	6,16 120,84,43	96,256,384,384,256 4096,4096,43	8,16,16,32,32 128,64,43
Activation Function	Tanh	ReLu	ReLu
Input Image Size	32x32x3	227x227x3	32x32x3
Noise	Not Added	Not Added	Not Added
Dropout	No	0.2	No
Class Number	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630

Table 4.3 Performance results of LeNet-5, AlexNet and DP1

	LeNet-5	AlexNet	Design Parameter1
Training Loss	0.3539	0.0952	0.2701
Training Accuracy	0.8888	0.9698	0.9132
Validation Loss	0.3759	0.2209	0.2442
Validation Accuracy	0.8816	0.9440	0.9279
Precision	0.9135	0.9552	0.9459
Recall	0.8582	0.9370	0.9158
F-Score	0.8846	0.9459	0.9304
Time	9 mins	30 hours	10.75 mins

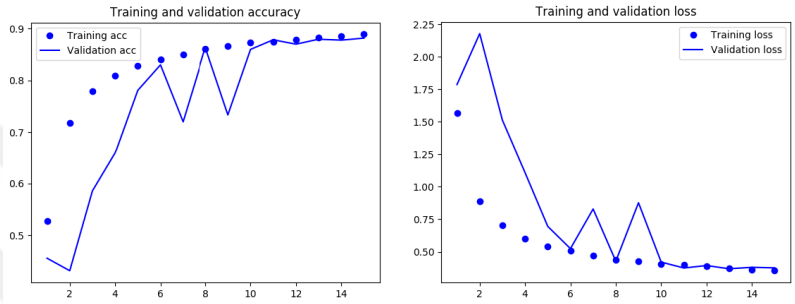

Figure 4.2 Graphical output of LeNet-5 architecture

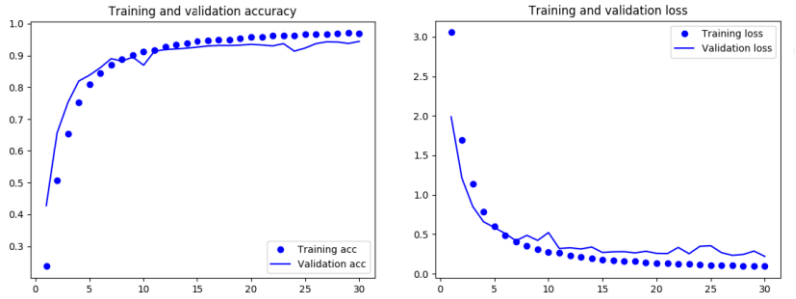

Figure 4.3 Graphical output of AlexNet architecture

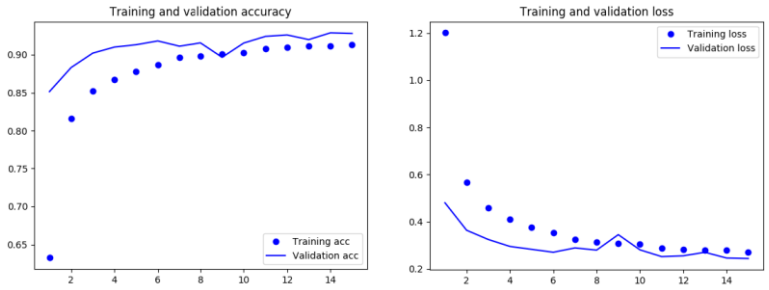

Figure 4.4 Graphical output of DP1

In order to obtain the results of the AlexNet architecture, the server mentioned under the heading Other Tools in Chapter 3 was used. Graphics, which is shown ram, disk and CPU usage during the process, were demonstrated in Figure 4.5, Figure 4.6 and Figure 4.7.

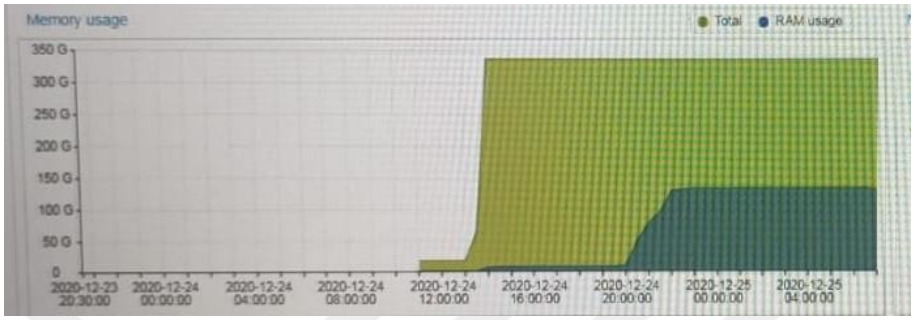

Figure 4.5 Illustration of RAM usage

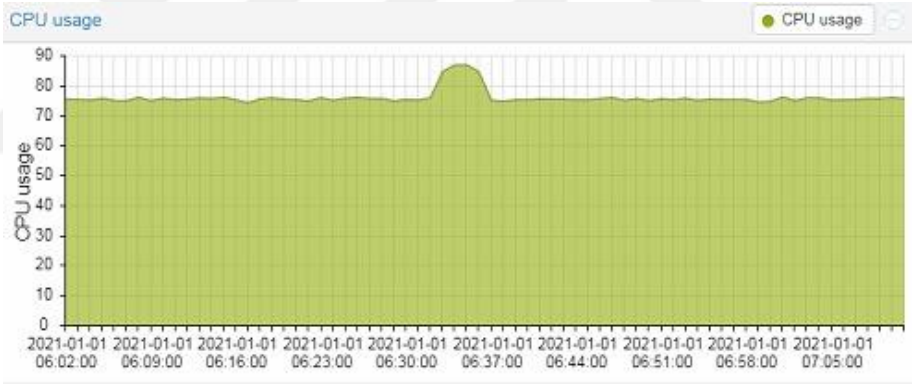

Figure 4.6 Illustration of CPU usage

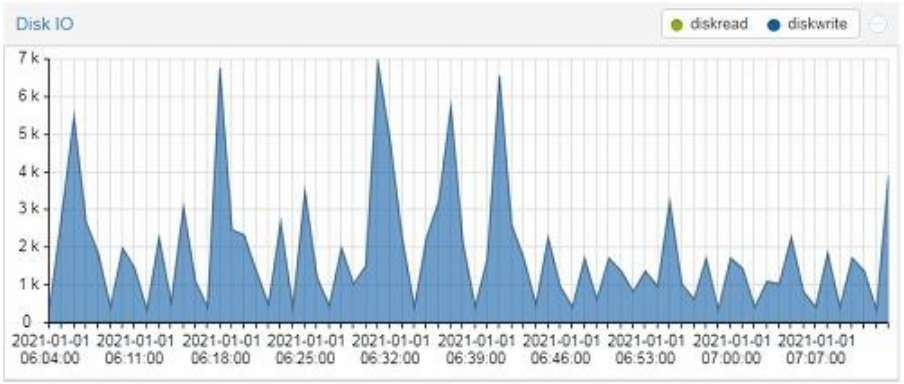

Figure 4.7 Illustration of disk usage

4.3 Created Design Parameters and Performance Analysis

After the reference design parameter was determined, other design parameters were created by making changes on the reference design parameter. Changes have been made on hyperparameters such as noise, depth and size of input image, activation function, train and test image numbers, dropout, learning rate, epoch number, batch size and filter numbers in the reference design parameter for performance analysis. The changes made on hyperparameters are shown in tables, the performance results also obtained were illustrated in tables and graphics.

4.3.1 Noise

Firstly, by adding noise to DP1, its effect on the result has been observed. Gaussian Noise is applicable in the Keras library. Therefore, this process was carried out with Gaussian noise. It is seen that 0.2 and 0.8 values are used in DP2 and DP3, respectively. This value, which can be chosen in range of 0 and 1, is specified as "stddev" in Keras Library. This float variable corresponds to the standard deviation.

When a data set is desired to be interpreted, all values are summed up and divided by the total number of data. Thus, the mean is obtained. However, in some cases it may not be accurate enough to comment only with mean. In such cases, more accurate evaluations can be made by using the standard deviation. The formula for standard deviation is as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \tilde{X})^2}{n-1}} \tag{4.5}$$

In this formula, "σ" correspond to standart deviation, "X_i" correspond to ith term in the data set, "$\widetilde{X}$" correspond to mean and "n" correspond to total number in the data set.

DP2 and DP3 were obtained as a result of the changes shown in Table 4.4. In consequence of these changes, the results in Table 4.5, Figure 4.8 and Figure 4.9 were acquired.

Table 4.4 Hyperparameter configurations of DP1, DP2 and DP3

	DP1	DP2	DP3
Learning Rate	0.01	0.01	0.01
Epoch	15	15	15
Batch Size	32	32	32
Filter Numbers	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32
	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3
Noise	**Not Added**	**Gaussian Noise (0.2)**	**Gaussian Noise (0.8)**
Dropout	No	No	No
Class Number	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630

Table 4.5 Performance results of DP1, DP2 and DP3

	DP1	DP2	DP3
Training Loss	0.2701	0.3632	0.5220
Training Accuracy	0.9132	0.8831	0.8286
Validation Loss	0.2442	0.2577	0.3202
Validation Accuracy	0.9279	0.9188	0.8991
Precision	0.9459	0.9386	0.9335
Recall	0.9158	0.9047	0.8725
F-Score	0.9304	0.9211	0.9015
Time	10.75 mins	9.25 mins	9.25 mins

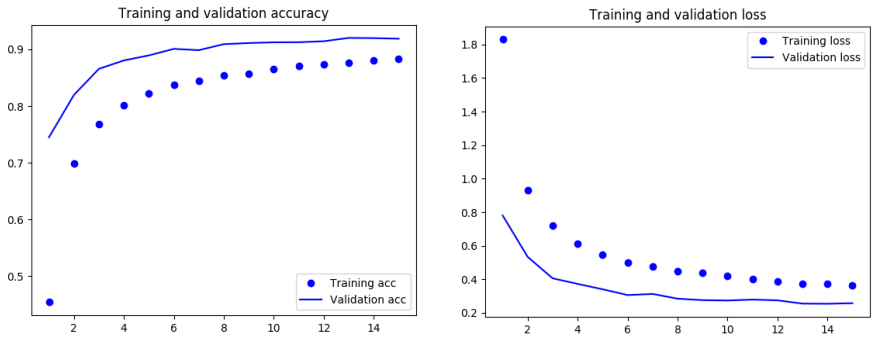

Figure 4.8 Graphical output of DP2

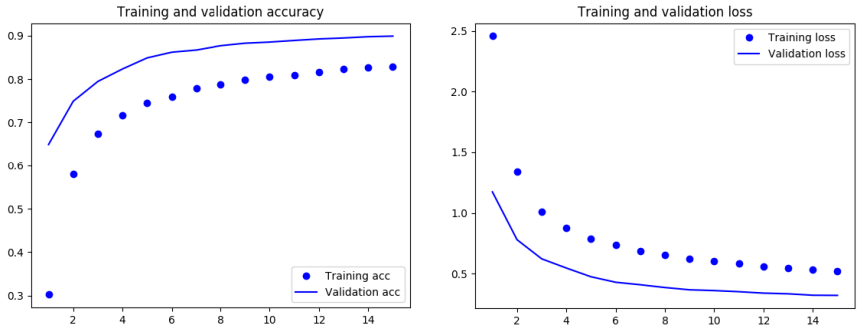

Figure 4.9 Graphical output of DP3

When the graphics and tables are examined, when noise is added to the images in the data set, loss increased and accuracy decreased. In this case, it can be stated that noise is a factor that negatively affects of models performance.

4.3.2 Depth and Size of Input Image

Nowadays, many pictures are used as three dimensions. However, with LeNet, which is the first successful CNN architecture, black and white images were processed. Therefore, all images in the data set were transformed into a gray scale and their effects on the result were examined. In addition, it was wanted to observe how the change in the size of the input image would reflect to the result. DP4, DP5 and DP6 were acquired by making modifications which is shown in Table 4.6 to demonstrate the effect of these changes on the result. As a result of the changes which are made, Table 4.7, Figure 4.10, Figure 4.11 and Figure 4.12 were obtained.

Table 4.6 Hyperparameter configurations of DP1, DP4, DP5 and DP6

	DP1	DP4	DP5	DP6
Learning Rate	0.01	0.01	0.01	0.01
Epoch	15	15	15	15
Batch Size	32	32	32	32
Filter Numbers	8,16,16,32,32	8,16,16,32,32	8,16,	8,16,16,
	128,64,43	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu	ReLu
Input Image Size	**32x32x3**	**32x32x1**	**10x10x3**	**64x64x3**
Noise	Not Added	Not Added	Not Added	Not Added
Dropout	No	No	No	No
Class Number	43	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630	39209/12630

Table 4.7 Performance results of DP1, DP4, DP5 and DP6

	DP1	DP4	DP5	DP6
Training Loss	0.2701	0.4242	0.9731	0.0941
Training Accuracy	0.9132	0.8691	0.6802	0.9711
Validation Loss	0.2442	0.7172	1.5465	0.1281
Validation Accuracy	0.9279	0.7861	0.5626	0.9632
Precision	0.9459	0.8431	0.6820	0.9737
Recall	0.9158	0.7538	0.4995	0.9580
F-Score	0.9304	0.7953	0.5751	0.9656
Time	10.75 mins	7 mins	4.8 mins	10 hours

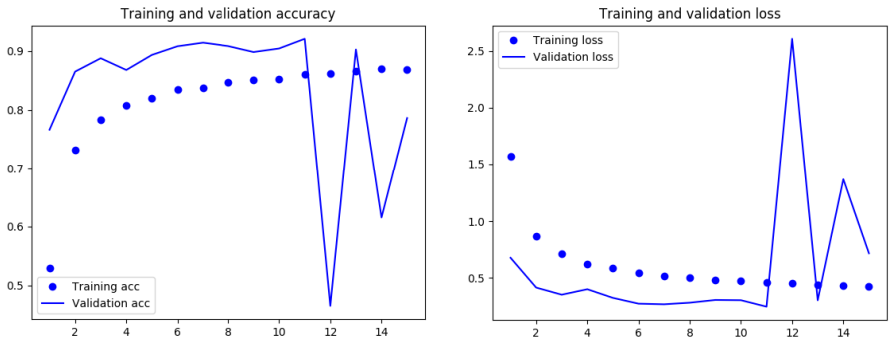

Figure 4.10 Graphical output of DP4

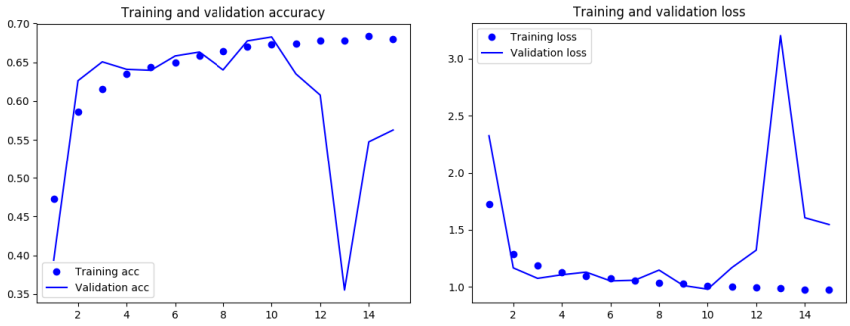

Figure 4.11 Graphical output of DP5

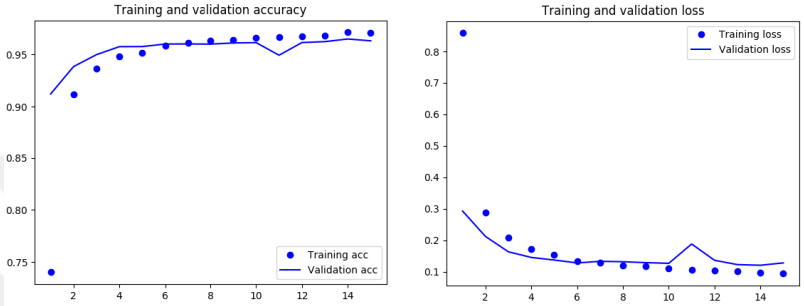

Figure 4.12 Graphical output of DP6

All pictures in the data set were made one dimensional so that the pictures are black and white. It is seen that performance decreases when DP1 and DP4 are compared. In other words, the gray scale of the input images affects the performance negatively for same data set. In DP5 and DP6, the size of the input image was changed. This situation caused a necessary change in the number of layers of the model. Diminishing the image size decreased the performance, while increasing the image size had an positive effect on the performance. Also, the server was used for DP6.

4.3.3 Activation Function

Softmax activation function was used at the output in the reference design parameter. ReLu was used in intermediate steps. In DP7 and DP8, as shown in Table 4.8, the effect of this situation on the result was examined by using different activation functions in the intermediate steps. As a result of the changes made, performance results in Table 4.9 have been procured. Graphical results of design parameters are shown in Figure 4.13 and Figure 4.14.

Table 4.8 Hyperparameter configurations of DP1, DP7 and DP8

	DP1	DP7	DP8
Learning Rate	0.01	0.01	0.01
Epoch	15	15	15
Batch Size	32	32	32
Filter Numbers	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32
	128,64,43	128,64,43	128,64,43
Activation Function	**ReLu**	**Tanh**	**Sigmoid**
Input Image Size	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added
Dropout	No	No	No
Class Number	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630

Table 4.9 Performance results of DP1, DP7 and DP8

	DP1	DP7	DP8
Training Loss	0.2701	0.4041	0.2173
Training Accuracy	0.9132	0.8721	0.9323
Validation Loss	0.2442	0.7689	0.2663
Validation Accuracy	0.9279	0.7830	0.9233
Precision	0.9459	0.8238	0.9415
Recall	0.9158	0.7551	0.9130
F-Score	0.9304	0.7874	0.9268
Time	10.75 mins	9 mins	9 mins

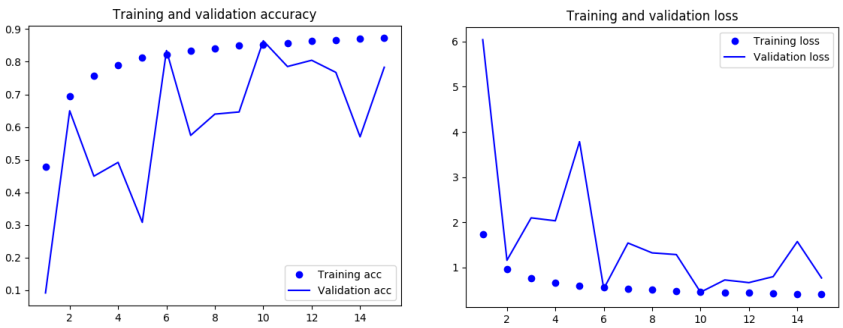

Figure 4.13 Graphical output of DP7

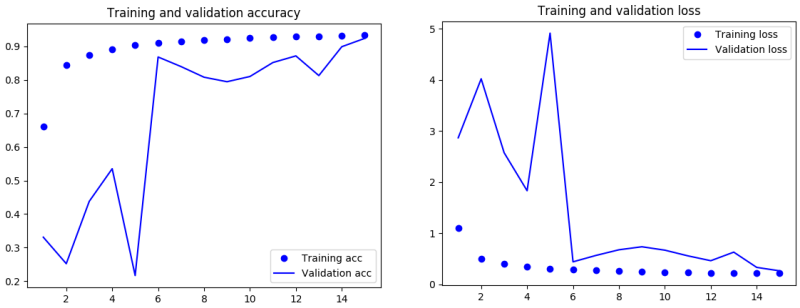

Figure 4.14 Graphical output of DP8

ReLU in DP1, tanh in DP7 and sigmoid activation function in DP8 were used. The best result was achieved with the ReLu activation function, while the worst result among the three was reached with the tanh activation function.

4.3.4 Train and Test Image Number

How many images the created algorithms are trained with is an important performance parameter. The effects of reducing the number of train images and the effects of adding a new class with different images were examined with DP9, DP10, DP11 and DP12 shown in Table 4.10. As a result of these changes, the results in Table 4.11, Figure 4.15, Figure 4.16, Figure 4.17 and Figure 4.18 were obtained.

Table 4.10 Hyperparameter configurations of DP1, DP9, DP10, DP11 and DP12

	DP1	DP9	DP10	DP11	DP12
Learning Rate	0.01	0.01	0.01	0.01	0.01
Epoch	15	15	15	15	15
Batch Size	32	32	32	32	32
Filter Numbers	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32
	128,64,43	128,64,43	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added	Not Added	Not Added
Dropout	No	No	No	No	No
Class Number	43	43	43	43	**44**
Train/ Test Image No	**39209/ 12630**	**29344/ 12630**	**19520/ 12630**	**9664/ 12630**	**39500/ 12680**

Table 4.11 Performance results of DP1, DP9, DP10, DP11 and DP12

	DP1	DP9	DP10	DP11	DP12
Training Loss	0.2701	0.2641	0.3222	0.2362	0.2824
Training Accuracy	0.9132	0.9158	0.8990	0.9254	0.9104
Validation Loss	0.2442	0.2668	0.3306	0.5233	0.2459
Validation Accuracy	0.9279	0.9244	0.8972	0.8458	0.9232
Precision	0.9459	0.9409	0.9271	0.8745	0.9413
Recall	0.9158	0.9144	0.8813	0.8269	0.9103
F-Score	0.9304	0.9273	0.9033	0.8496	0.9253
Time	10.75 mins	8.3 mins	5 mins	3.25 mins	9.75 mins

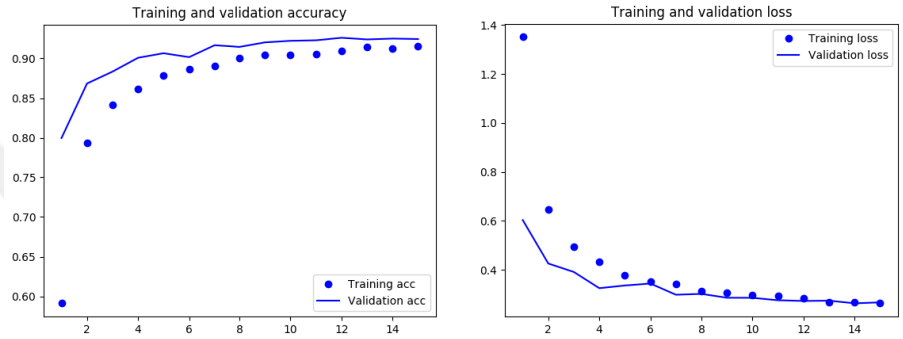

Figure 4.15 Graphical output of DP9

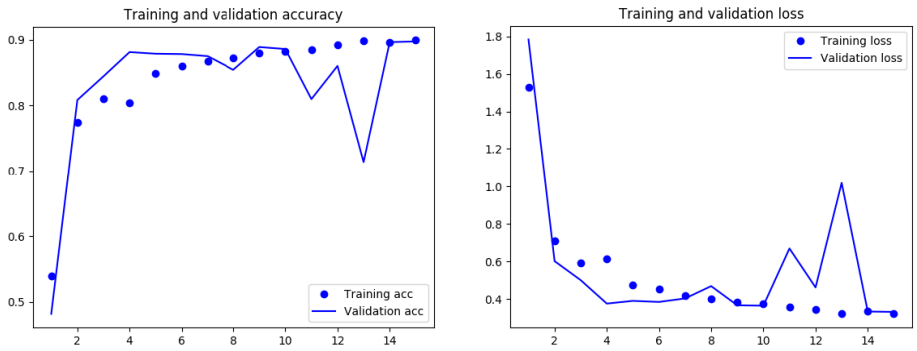

Figure 4.16 Graphical output of DP10

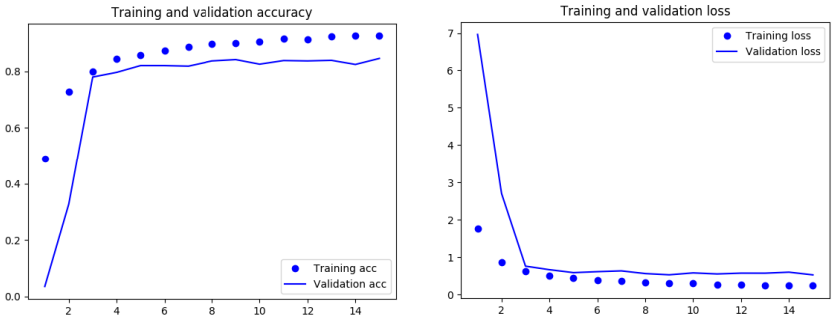

Figure 4.17 Graphical output of DP11

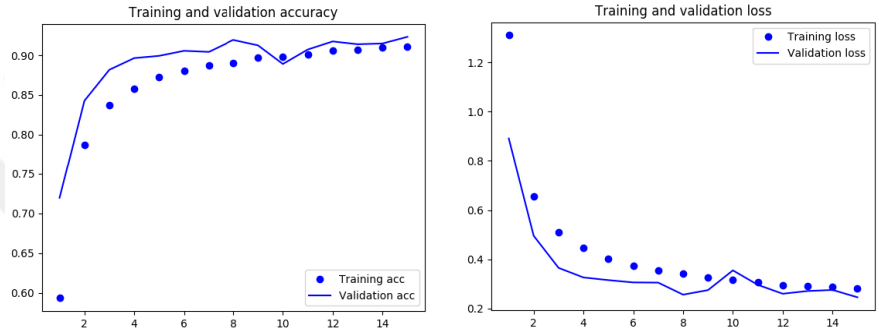

Figure 4.18 Graphical output of DP12

The GTSD used in this study contains over fifty thousand image data. Approximately forty thousand of these pictures are separated as training data and ten thousand as test data. The number of training images in DP9, DP10 and DP11 have been reduced by 25%, 50% and 75%, respectively. With the decrease in the number of pictures that the design parameters used to train themselves, the performance obtained from the models decreased. In DP12, a new class has been added to the data set. Therefore, the number of training and test images has been increased. This situation did not influence the performance of the system much.

4.3.5 Dropout

Hyperparameter adjustments specified in Table 4.12 were made to examine the effects of dropout layer which is also used in AlexNet architecture. AlexNet with

dropout layer achieved great successful in 2012. The results of DP13, DP14 and DP16 are shown in Table 4.13, Figure 4.19, Figure 4.20 and Figure 4.21.

Table 4.12 Hyperparameter configurations of DP1, DP13, DP14 and DP15

	DP1	DP13	DP14	DP15
Learning Rate	0.01	0.01	0.01	0.01
Epoch	15	15	15	15
Batch Size	32	32	32	32
Filter Numbers	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32
	128,64,43	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added	Not Added
Dropout	**No**	**0.2**	**0.4**	**0.8**
Class Number	43	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630	39209/12630

Table 4.13 Performance results of DP1, DP13, DP14 and DP15

	DP1	DP13	DP14	DP15
Training Loss	0.2701	0.3163	0.5544	0.6992
Training Accuracy	0.9132	0.8978	0.8265	0.7775
Validation Loss	0.2442	0.3249	0.3792	0.3900
Validation Accuracy	0.9279	0.9030	0.8788	0.8673
Precision	0.9459	0.9250	0.9201	0.9191
Recall	0.9158	0.8922	0.8558	0.8317
F-Score	0.9304	0.9081	0.8861	0.8724
Time	10.75 mins	9 mins	9 mins	9 mins

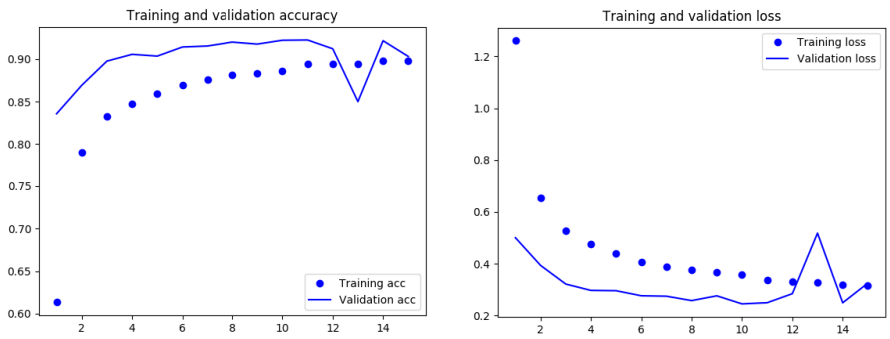

Figure 4.19 Graphical output of DP13

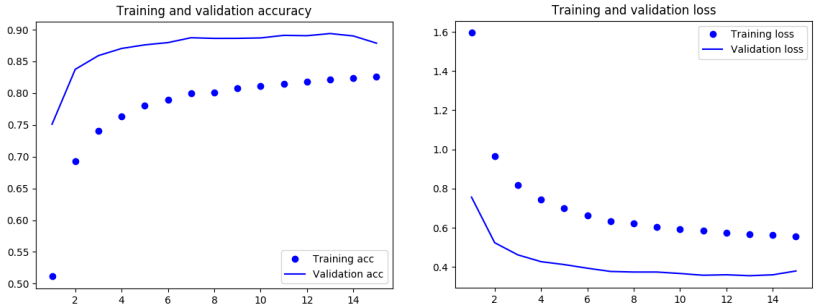

Figure 4.20 Graphical output of DP14

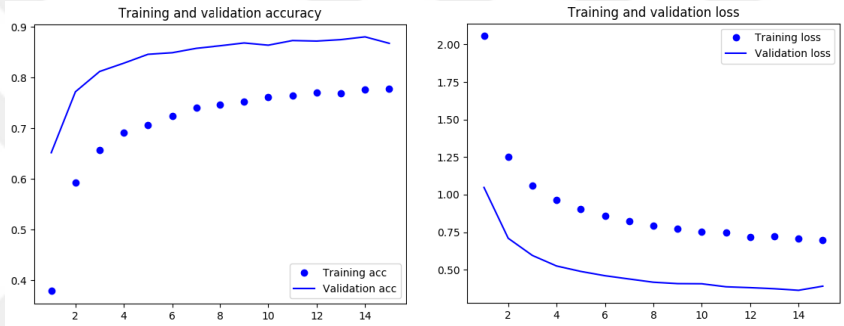

Figure 4.21 Graphical output of DP15

Using dropout is a simple and effective method so that the model does not have overfitting. It becomes a very important hyperparameter, especially in very large data sets. Since the data set used in this work is not very large, dropout is not used in the reference design parameter. It is also important at what rate to use dropout. In this section, as the ratio goes up, the decrease in the performance of the model increases. When the dropout value selects greater than 0.5, a warning is received by Keras. For example, in the case where 0.8 was selected, a warning was given: “Large dropout rate: 0.8 (> 0.5). Please ensure that this is intended. ”

4.3.6 Learning Rate

Another hyperparameter that has a great impact on performance is learning rate. It can be considered as the step size to be taken. The changes shown in Table 4.14 were made and how much this hyperparameter affected the result was examined. In

addition, in order to better observe the effect of the result, the number of epoch was chosen as 120 in all design parameters. Performance results of DP16, DP17, DP18, DP19, DP20 and DP21 are shown in Table 4.15, Figure 4.22, Figure 4.23, Figure 4.24, Figure 4.25, Figure 4.26 and Figure 4.27.

Table 4.14 Hyperparameter configurations of DP16, DP17, DP18, DP19, DP20 and DP21

	DP16	DP17	DP18	DP19	DP20	DP21
Learning Rate	**1**	**0.5**	**0.1**	**0.05**	**0.01**	**0.001**
Epoch	120	120	120	120	120	120
Batch Size	32	32	32	32	32	32
Filter Numbers	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32	8,16,16,32,32
	128,64,43	128,64,43	128,64,43	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added	Not Added	Not Added	Not Added
Dropout	No	No	No	No	No	No
Class Number	43	43	43	43	43	43
Train/ Test Image No	39209/ 12630	39209/ 12630	39209/ 12630	39209/ 12630	39209/ 12630	39209/ 12630

Table 4.15 Performance results of DP16, DP17, DP18, DP19, DP20 and DP21

	DP16	DP17	DP18	DP19	DP20	DP21
Training Loss	0.4810	0.2689	0.2120	0.1705	0.1384	0.1369
Training Accuracy	0.8449	0.9124	0.9325	0.9444	0.9553	0.9557
Validation Loss	2703	1981	0.2152	0.1874	0.1826	0.1805
Validation Accuracy	0.8897	0.8936	0.9349	0.9434	0.9481	0.9500
Precision	0.9173	0.9112	0.9504	0.9558	0.9549	0.9584
Recall	0.8686	0.8851	0.9264	0.9385	0.9429	0.9465
F-Score	0.8919	0.8977	0.9380	0.9470	0.9488	0.9523
Time	36.5 mins	40.5 mins	48.5 mins	54.5 mins	58.5 mins	72.5 mins

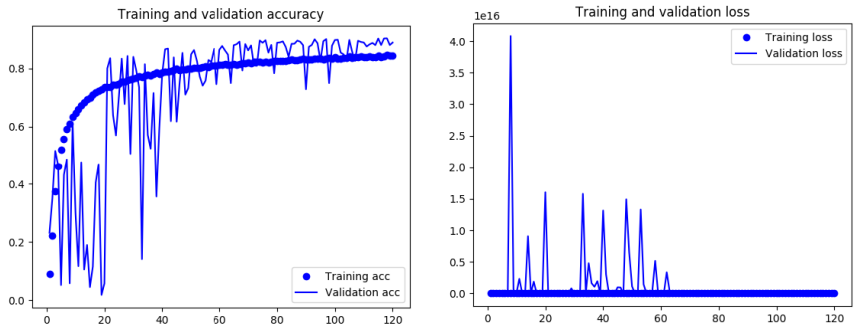

Figure 4.22 Graphical output of DP16

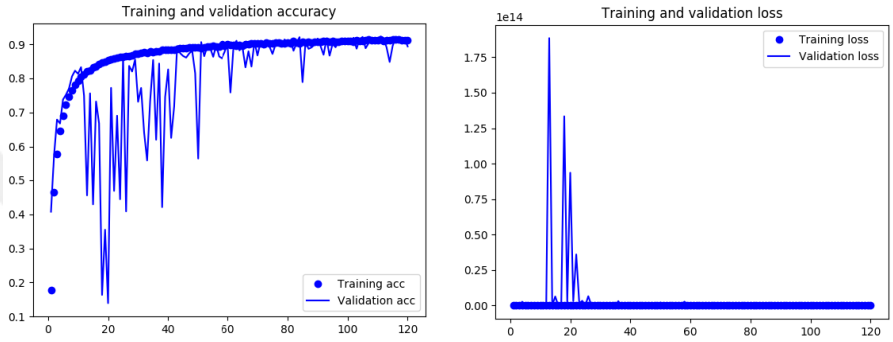

Figure 4.23 Graphical output of DP17

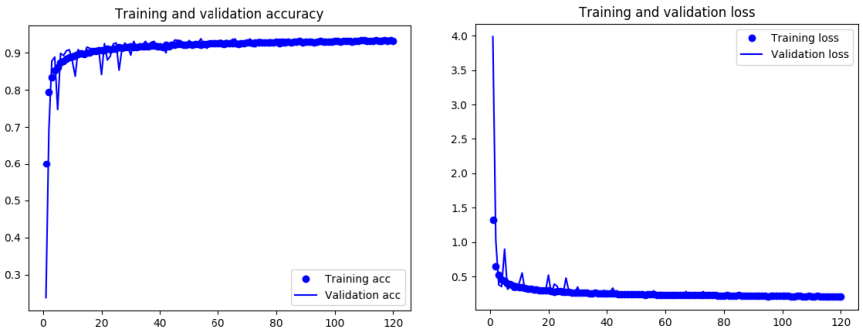

Figure 4.24 Graphical output of DP18

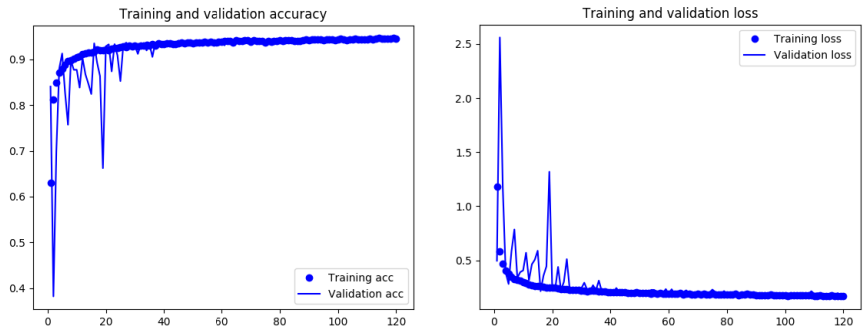

Figure 4.25 Graphical output of DP19

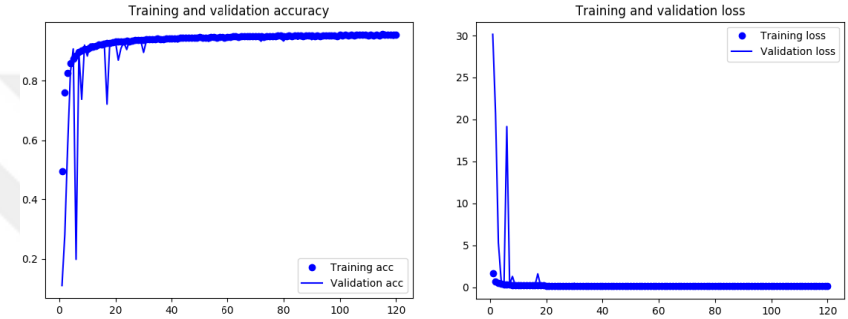

Figure 4.26 Graphical output of DP20

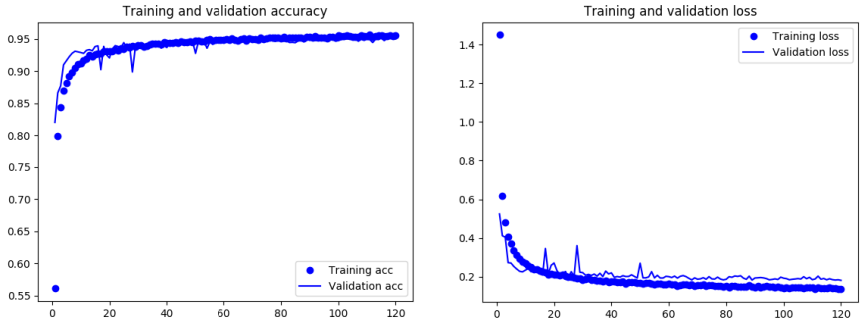

Figure 4.27 Graphical output of DP21

While examining the effect of learning rate on the result, epoch number was chosen as 120, unlike other design parameters. Because when the epoch value was selected as 15, worse results were obtained for low learning rate values. When this change was made, like shown in Table 4.15, as the learning rate decreased, the performance

of the model increased. As the value decreases, the training time of the model goes up, while the great zig-zags seen in validation accuracy and validation loss graphs diminished.

4.3.7 Epoch Number

Epoch number represents how many times the entire data set of the algorithm has been seen. It is also called as loop in some sources. Four different design parameters have been derived by making the changes on hyperparameter as shown in Table 4.16. Numerical performance results obtained after the created design parameters are shown in Table 4.17 and graphical results are shown in Figure 4.28, Figure 4.29, Figure 4.30 and Figure 4.31.

Table 4.16 Hyperparameter configurations of DP1, DP22, DP23, DP24 and DP25

	DP1	DP22	DP23	DP24	DP25
Learning Rate	0.01	0.01	0.01	0.01	0.01
Epoch	**15**	**5**	**30**	**60**	**120**
Batch Size	32	32	32	32	32
Filter Numbers	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32
	128,64,43	128,64,43	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added	Not Added	Not Added
Dropout	No	No	No	No	No
Class Number	43	43	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630	39209/12630	39209/12630

Table 4.17 Performance results of DP1, DP22, DP23, DP24 and DP25

	DP1	DP22	DP23	DP24	DP25
Training Loss	0.2701	0.2767	0.2336	0.1857	0.1203
Training Accuracy	0.9132	0.9117	0.9246	0.9405	0.9616
Validation Loss	0.2442	0.4676	0.2064	0.1950	0.1473
Validation Accuracy	0.9279	0.8663	0.9352	0.9417	0.9554
Precision	0.9459	0.9012	0.9520	0.9533	0.9645
Recall	0.9158	0.8472	0.9275	0.9349	0.9497
F-Score	0.9304	0.8729	0.9394	0.9439	0.9569
Time	10.75 mins	6.25 mins	18 mins	27 mins	36 mins

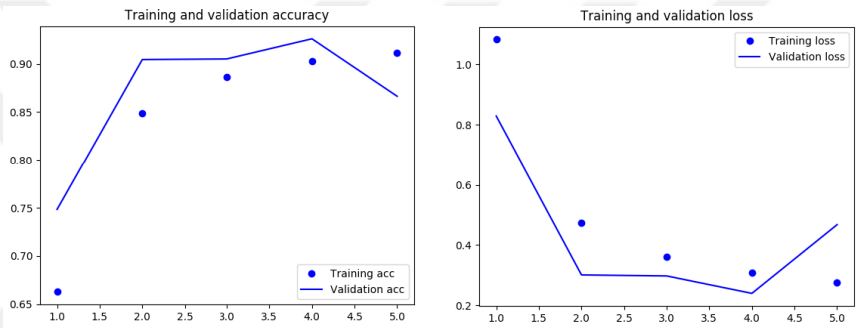

Figure 4.28 Graphical output of DP22

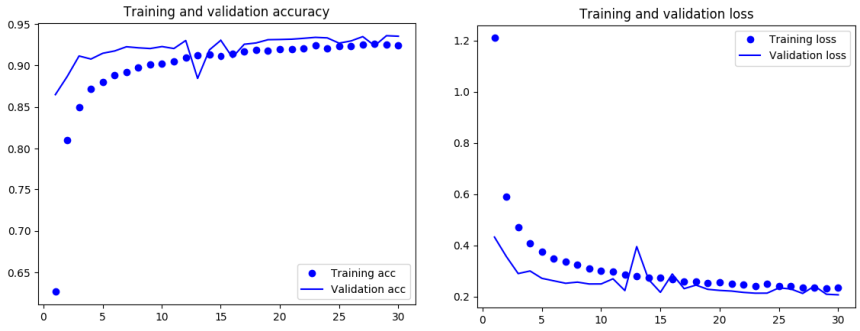

Figure 4.29 Graphical output of DP23

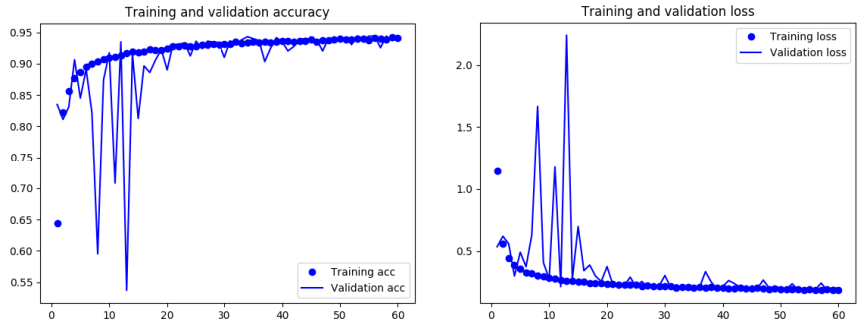

Figure 4.30 Graphical output of DP24

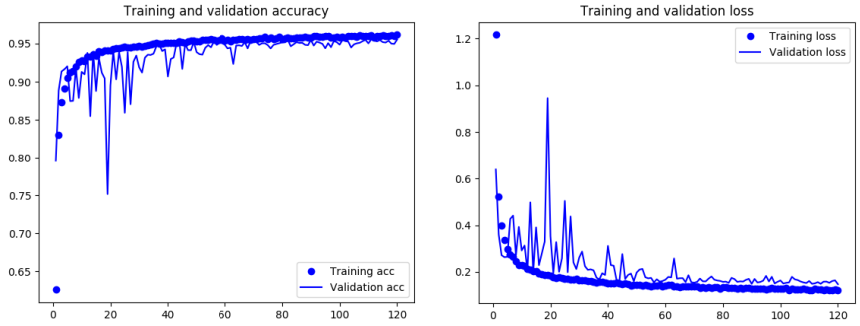

Figure 4.31 Graphical output of DP25

The increase in the number of epochs allows the model to train itself better. For this reason, considering the values in Table 4.17, as the epoch number increases; the accuracy, precision, recall, f-score, time values have gradually increased, and the loss value is gradually diminished.

4.3.8 Batch Size

Taking into account the performance of the device used during the process, it would make sense to split the data set into parts. This structure is called mini-batch. Hyperparameter configuration for DP26, DP27, DP28 and DP29 is shown in Table 4.18. As a result of the changes shown in the table, the results demonstrated in Table 4.19, Figure 4.32, Figure 4.33, Figure 4.34 and Figure 4.35 were obtained.

Table 4.18 Hyperparameter configurations of DP1, DP26, DP27, DP28 and DP29

	DP1	DP26	DP27	DP28	DP29
Learning Rate	0.01	0.01	0.01	0.01	0.01
Epoch	15	15	15	15	15
Batch Size	**32**	**2**	**8**	**128**	**256**
Filter Numbers	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32	8,16,16, 32,32
	128,64,43	128,64,43	128,64,43	128,64,43	128,64,43
Activation Function	ReLu	ReLu	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added	Not Added	Not Added
Dropout	No	No	No	No	No
Class Number	43	43	43	43	43
Train/ Test Image No	39209/ 12630	39209/ 12630	39209/ 12630	39209/ 12630	39209/ 12630

Table 4.19 Performance results of DP1, DP26, DP27, DP28 and DP29

	DP1	DP26	DP27	DP28	DP29
Training Loss	0.2701	3.3061	0.8167	0.1400	0.1803
Training Accuracy	0.9132	0.1170	0.7446	0.9553	0.9421
Validation Loss	0.2442	4.0423	0.4890	0.2406	0.2030
Validation Accuracy	0.9279	0.1316	0.8504	0.9347	0.9397
Precision	0.9459	0.0263	0.9201	0.9451	0.9533
Recall	0.9158	0.0136	0.8014	0.9280	0.9318
F-Score	0.9304	0.0178	0.8526	0.9364	0.9423
Time	10.75 mins	22 mins	13.5 mins	10.75 mins	9.75 mins

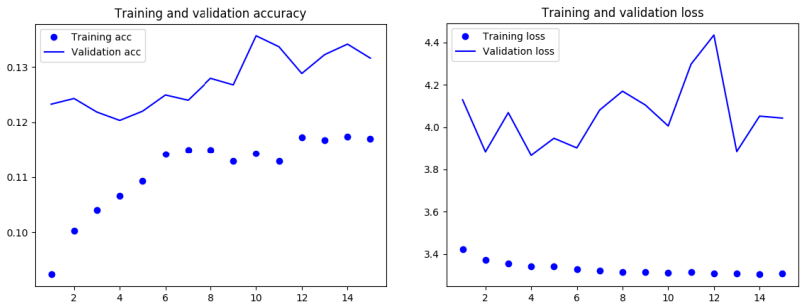

Figure 4.32 Graphical output of DP26

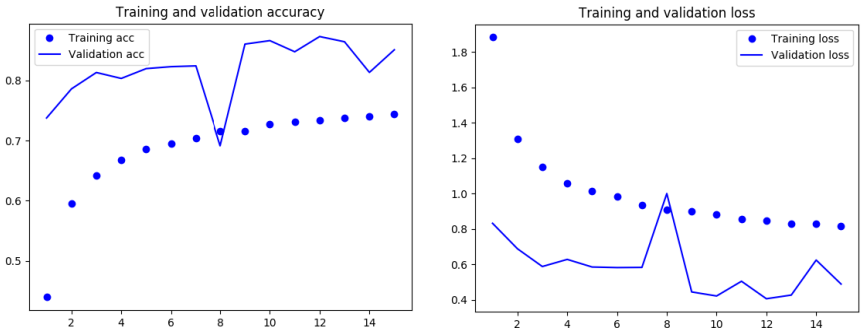

Figure 4.33 Graphical output of DP27

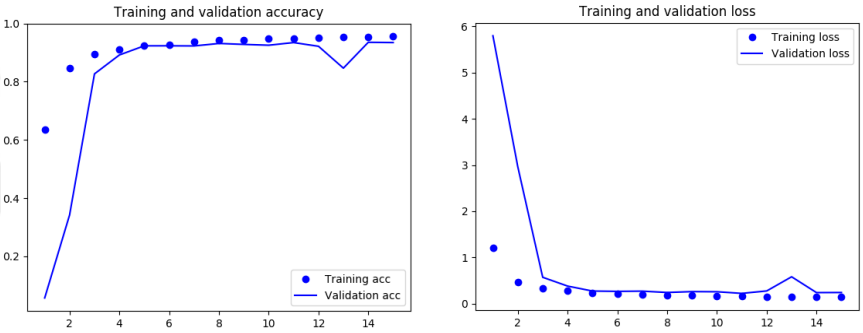

Figure 4.34 Graphical output ofDP28

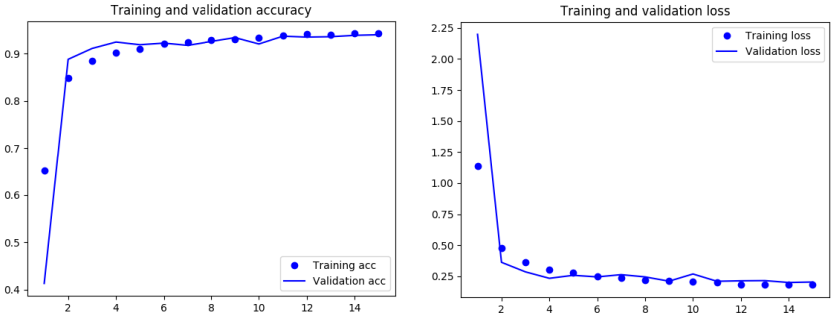

Figure 4.35 Graphical output of DP29

There are 39209 training pictures in GTSD. If you choose batch size as one, the device will do the learning process for all the pictures one by one. However, if thirty two is selected as batch size like in the reference algorithm, an epoch will be completed at the end of 1225 iterations. Because in this case, the data set will be divided into 1225 groups, each containing 32 pictures. Batch size values have been

changed to 2, 8, 128 and 256 for DP26, DP27, DP28, and DP29, respectively. After these changes, it has been observed that the go up in batch size increases the performance of the model up to a certain point. This increase also contributes positively to the performance of the device used. In addition, as the batch size goes up since the number of pictures in the group increases, the model can train itself more accurately. Having the opportunity to see more different image examples in each group provides a better generalization for model. Therefore, the fluctuation in the graphics is gradually decreasing.

4.3.9 Filter Number

The features of the pictures in the data set are extracted with the help of filters. Therefore, the number of filters used is also an important hyperparameter. In addition, the number of nodes in the fully-connected layer is also a parameter that can be changed while creating the algorithm. Hyperparameter values of DP30, DP31, DP32 and DP33 are shown in Table 4.20. After the design parameters were adjusted as shown in the table below, the results in Table 4.21, Figure 4.36, Figure 4.37, Figure 4.38, Figure 4.39 were procured.

Table 4.20 Hyperparameter configurations of DP1, DP30, DP31, DP32 and DP33

	DP1	DP30	DP31	DP32	DP33
Learning Rate	0.01	0.01	0.01	0.01	0.01
Epoch	15	15	15	15	15
Batch Size	32	32	32	32	32
Filter Numbers	**8,16,16, 32,32**	**4,8,8, 16,16**	**64,32,32, 128,128**	8,16,16,32,32	8,16,16,32,32
	128,64,43	128,64,43	128,64,43	**16,8,43**	**1024,512,43**
Activation Function	ReLu	ReLu	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	32x32x3	32x32x3	32x32x3
Noise	Not Added	Not Added	Not Added	Not Added	Not Added
Dropout	No	No	No	No	No
Class Number	43	43	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630	39209/12630	39209/12630

Table 4.21 Performance results of DP1, DP30, DP31, DP32 and DP33

	DP1	DP30	DP31	DP32	DP33
Training Loss	0.2701	3.4861	0.0578	0.4073	0.2113
Training Accuracy	0.9132	0.0574	0.9818	0.8725	0.9312
Validation Loss	0.2442	17688	0.1952	0.3205	0.2374
Validation Accuracy	0.9279	0.0285	0.9470	0.9028	0.9306
Precision	0.9459	0.0285	0.9581	0.9376	0.9499
Recall	0.9158	0.0285	0.9420	0.8817	0.9216
F-Score	0.9304	0.0285	0.9499	0.9084	0.9553
Time	10.75 mins	9 mins	17 mins	9 mins	9.75 mins

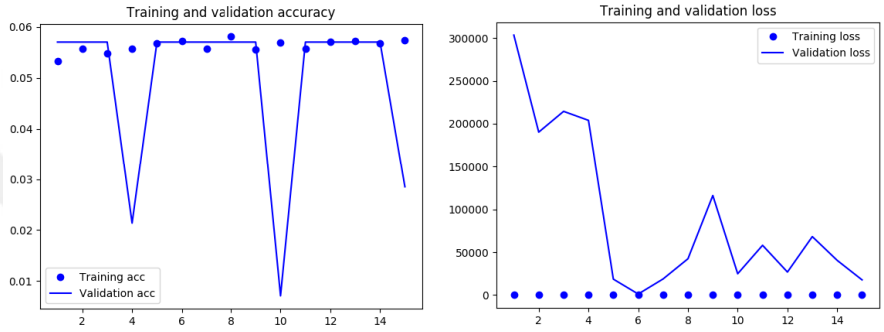

Figure 4.36 Graphical output of DP30

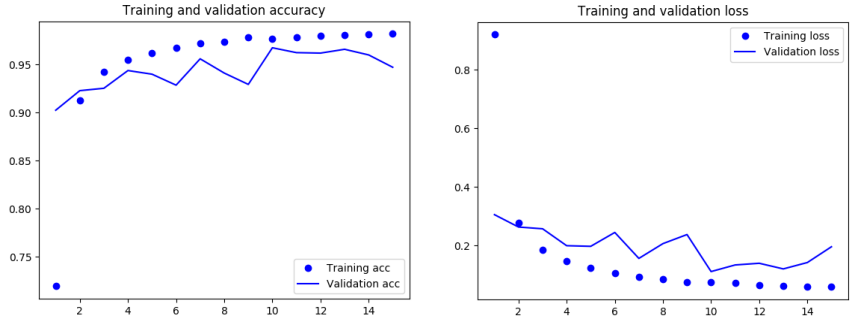

Figure 4.37 Graphical output of DP31

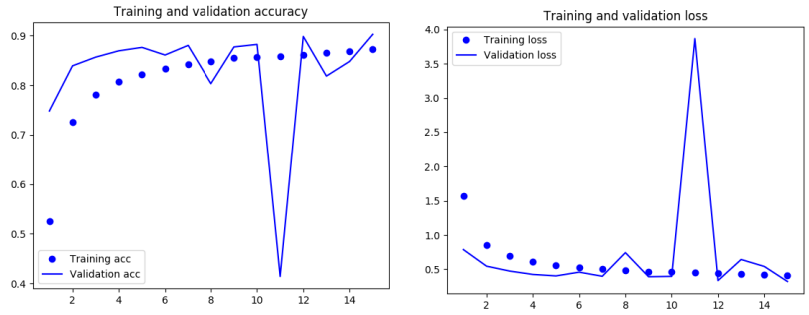

Figure 4.38 Graphical output of DP32

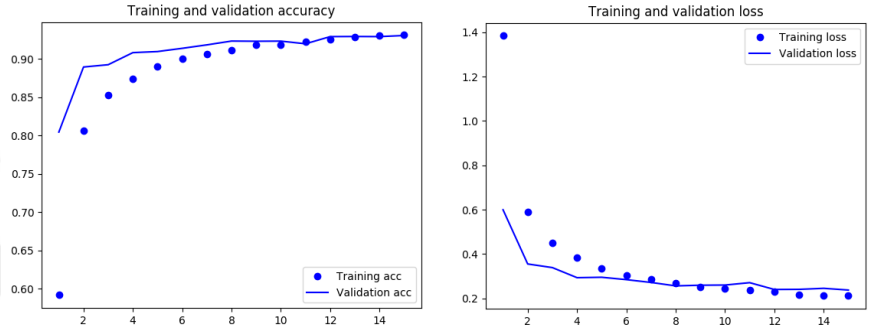

Figure 4.39 Graphical output of DP33

While the number of filters used in the convolutional layers was decreased in DP30, the number of these filters was increased in DP31. As more filters provide more feature extraction, the quality of self-training of the model goes up. When DP30 and DP31 compared to DP1, lower results obtained with DP30 and higher results acquired with DP31. In DP32 and DP33, the effect of the change of the number of nodes in the fully-connected layer on performance has been examined. While decreasing this value has negative effect on the performance, increasing it has positive effect on the performance.

4.3.10 Maximum Performance

After the performance results obtained with other design parameters, two different design parameters were finally created. Using slightly different configuration settings, we tried to achieve the highest performance results. For this, the values specified in the table in Table 4.22 are used. With the design parameters created, the results visualized in Table 4.23, Figure 4.40 and Figure 4.41 were reached.

Table 4.22 Hyperparameter configurations of DP1, DP34 and DP35

	DP1	**DP34**	**DP35**
Learning Rate	0.01	0.001	0.01
Epoch	15	15	15
Batch Size	32	32	128
Filter Numbers	8,16,16,32,32	64,64,64,128,128,	64,128,128,
	128,64,43	1024,1024,43	1024,1024,43
Activation Function	ReLu	ReLu	ReLu
Input Image Size	32x32x3	32x32x3	64x64x3
Noise	Not Added	Not Added	Not Added
Dropout	No	No	0.2
Class Number	43	43	43
Train/ Test Image No	39209/12630	39209/12630	39209/12630

Table 4.23 Performance results of DP1, DP34 and DP35

	DP1	DP34	DP35
Training Loss	0.2701	0.0653	0.0037
Training Accuracy	0.9132	0.9801	0.9988
Validation Loss	0.2442	0.1395	0.0879
Validation Accuracy	0.9279	0.9624	0.9751
Precision	0.9459	0.9705	0.9791
Recall	0.9158	0.9553	0.9734
F-Score	0.9304	0.9627	0.9762
Time	10.75 mins	20 mins	10.2 hours

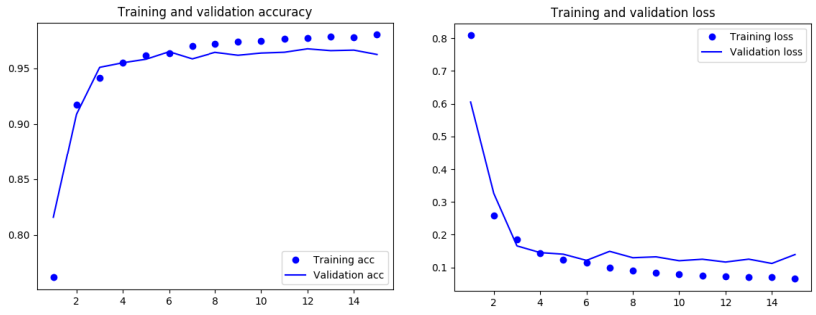

Figure 4.40 Graphical output of DP34

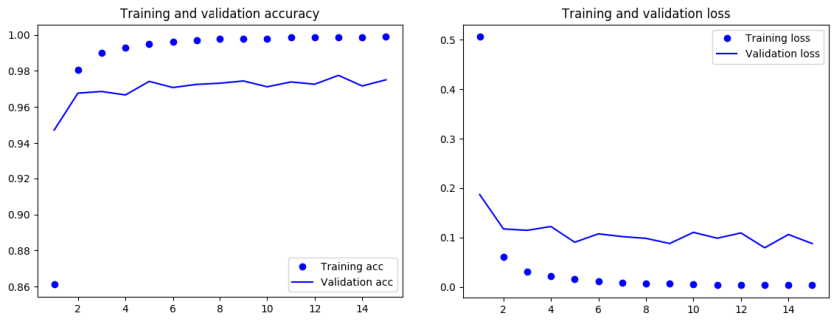

Figure 4.41 Graphical output of DP35

In line with the results observed in other design parameters, it has been tried to attain maximum performance in DP34 and DP35. For this, different values have been selected for learning rate, batch size, input image size and dropout hyperparameters. Both of the models configured with different values determined, reached approximately 97% accuracy. For DP34, Acer PC was used and the process took twenty minutes. DP35 was trained on the server and the process took approximately ten hours.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Deep learning has become very popular in recent years. It is frequently encountered as an indispensable term that is effectively used in many fields such as education, health, transportation, space-science. In this study, the CNN technique, which gained popularity again after the success in ILSVRC 2012, was used. CNN has accomplished many challenging object detection tasks. The main purpose of this master thesis is to perform object detection using convolutional neural network. While applying this technique, obtaining maximum performance is the most important criterion. For this, it is necessary to make some configuration settings called hyperparameters, which are made while creating the model. In this master thesis, thirty-five different design parameters were created by changing hyperparameters such as noise, depth and size of input image, activation function, train and test image numbers, dropout, learning rate, epoch number, batch size and filter numbers. Considering that LeNet-5 and AlexNet are also implemented, the results obtained from thirty-seven algorithms in total were compared. Then the obtained results were presented by visualizing them with tables and graphics.

When the tables and graphics are examined; it has been observed that noise, converting images to grayscale form, reducing the size of the input image, using tanh or sigmoid activation functions instead of ReLU, reducing the number of train images, and using a high dropout value decrease the performance of the model. However, it was observed that while decreasing the learning rate value, increasing the number of epochs, batch size value, number of filters used in the convolution layer and number of nodes in the fully-connected layer positively affected the performance of the model. Finally, DP34 and DP35 are derived and two high-performance models are presented in line with the observations made from the performance results of other design parameters. In addition, it has been shown that

similar high results can be obtained for different design parameters with these models.

When the collective tables which is given in appendix section are examined, it can be seen that with the correct hyperparameter settings, accuracy, precision, recall and f-score values of around 97% were obtained. Time expressions with "*" sign indicate that the operation of the related design parameter is done with the server whose technical features are explained in Section 3.7.4 “Other Tools”.

Many trials have been made for each hyperparameter and appropriate results have been presented. However, the trial process at this stage was quite time consuming. For this reason, not all hyperparameters that would be observed are included. In further studies; hyperparameters such as optimizer, loss function, kernel size, stride, padding, pooling method are also thought to be added to the work. New design parameters will be created with the additions planned. It is planned to apply all design parameters, including the new design parameters to be obtained, to another datasets. Thus, a more general comparison table will be acquired.

REFERENCES

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in Artificial Neural Network Applications: A Survey. *Heliyon* **4**.

Albawi, S., Mohammed, T. A., & Al-Zavi, S. (2017). Understanding of a Convolutional Neural Network. *International Conference on Engineering and Technology (ICET)*, 1-6. Antalya.

Alom, Z., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, S., Asari , V. (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. *ArXiv*.

Ayumi, V., Rere, R., Fafany, M., & Arymurti, A. (2016). Optimization of Convolutional Neural Network using Microcanonical Annealing Algorithm. *International Conference of Advanced Computer Science and Information Systems (ICACSIS).*

Barmettler, J. (2019). *Hackernoon*. https://hackernoon.com/the-full-story-behind-convolutional-neural-networks-and-the-math-behind-it-2j4fk3zu2 .

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 1-55.

Bengio, Y., Goodfellow, I., & Courville, A. (2016). *Deep learning.* Cambridge, MA, USA: The MIT Press.

Bergstra, J., Yamins, D., & Cox, D. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 115-123. Atlanta, Georgia, USA,.

Bochinski, E., Senst, T., & Sikora, T. (2017). Hyper-parameter Optimization for Convolutional Neural Network Committees Based on Evolutionary Algorithms. *2017*

IEEE International Conference on Image Processing (ICIP), 3924-3928. Beijing, China.

Brownlee, J. (2019). *Deep Learning for Computer Vision: Image Classificiation, Object Detection and Face Recognition in Python.* Machine Learning Mastery.

Brutus, M. (2018). *Pinterest.* https://tr.pinterest.com/pin/348114246175914595/ .

Cass, S. (2020). *IEEE Spectrum.* https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020?referrer=%2F .

Chandra, A. (2018). *McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron.* Towards Data Science: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

Changzhen, X., Cong, W., Weixin, M., & Yanmei, S. (2016). A Traffic Sign Detection Algorithm Based on Deep Convolutional Neural Network. *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, 676-679.

Chen, W. (2020). *Medium.* https://williamjchen.medium.com/the-one-stop-guide-to-convolutional-neural-networks-2a6e81de1d59

Ciresan , D., Gambardella, L. M., Giusti, A., & Schmidhuber, J. (2012). Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. *Proceedings of Neural Information Processing Systems* **25**.

Ciresan, D., Ueli, M., Masci, J., & Schmidhuber, J. (2012). Multi-Column Deep Neural Network for Traffic Sign Classification. *Neural networks : The Official Journal of the International Neural Network Society* **32**, 333-8.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 273-297.

Dan, C., Gambardella, L., Meier, E., & Schmidhuber, J. (2011). Convolutional Neural Network Committees For Handwritten Character Classification. *2011 International Conference on Document Analysis and Recognition*, 1135-1139. Beijing,China.

Data Hacker. (2018). http://datahacker.rs/deep-learning-alexnet-architecture/ .

Deng, L., & Yu, D. (2014). *Deep Learning: Methods and Applications.* Now Foundations and Trends.

Dettmers, T. (2015). *Deep Learning in a Nutshell: History and Training.* Nvidia: https://developer.nvidia.com/blog/deep-learning-nutshell-history-training/ .

Devero, A. (2015). *Blog Alex Devero*. https://blog.alexdevero.com/programming-languages-libraries-and-frameworks/ .

Dumoulin, V., & Visin, F. (2016). A Guide to Convolution Arithmetic for Deep Learning. *ArXiv*.

Fukushima , K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 193-202.

Gao, C., Hayward, R., & Müller, M. (2018). Move Prediction Using Deep Convolutional Neural Networks in Hex. *IEEE Transactions on Games* **10**, 336-343.

Gershenson, C. (2003). *Artificial Neural Networks for Beginners*.

Girshick, R., Donahue, J., Darrell, T., Malik, J., & Berkeley, U. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580-587. Columbus.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Google. (2021). *Google.* Google: trends.google.com

Gupta, N. (2013). Artificial Neural Network. *International Conference on Recent Trends in Applied Sciences with Engineering Applications.*

Gülcü , A., & Kuş, Z. (2019). A Survey of Hyper-parameter Optimization Methods in Convolutional Neural. *Gazi Üniversitesi Fen Bilimleri Dergisi*, 503-522.

Hale, J. (2018). *Towards Data Science*. https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a .

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778 . Las Vegas,NV, USA.

Hinton, G. (2007). Learning Multiple Layers of Representation. *Trends in Cognitive Sciences* **11**, 428-34.

Hinton, G., Dayan, P., Frey, B., & Neal, R. (1995). The Wake-Sleep Algorithm for Unsupervised. *Science* **268**, 1158-1161.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-term Memory. *Neural Computation* **9**, 1735-80.

Hopfield, J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, 2554-2558.

Huang, T. (1996). Computer Vision: Evolution and Promise. *CERN School of computing*, 21-26. Egmond aan Zee.

Hubel , D., & Wiesel , T. (1959). Receptive Fields of Single Neurones in the Cat's Striate Cortex. *The Journal of Physiology*, 574-591.

Ivakhnenko, A., & Lapa, V. (1966). *Cybernetic Predicting Devices.* Washington: Joint Publications Research Service.

Jeffcock, P. (2018). Oracle: https://blogs.oracle.com/bigdata/difference-ai-machine-learning-deep-learning .

Joshi, R. (2016). *Exsilio Blog*. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ .

Karami, E., Prasad, S., & Shehata, M. (2015). Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. *Newfoundland Electrical and Computer Engineering Conference.* St. John's.

Kelley, H. (1960). Gradient Theory of Optimal Flight Paths. *ARS Journal*, 947-954.

Keras. (2021). https://keras.io/about/ .

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems* **25**.

Kurt, F. (2018). Evrişimli Sinir Ağlarinda Hiper Parametrelerin Etkisinin İncelenmesi. Ankara, Türkiye.

Kwak, N. (2016). Introduction to Convolutional Neural Networks(CNNs). Korea.

LeCun, Y., Bengio , Y., & Hinton , G. (2015). Deep learning. *Nature* **521**, 436-444.

Lecun, Y., Boser, B., Denker , J., Henderson, D., Howard, R., & Hubbard , W. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 541-551.

Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, w., & Jackel, L. D. (1989). Handwritten Digit Recognition with a Back-Propagation Network.

Lecunn, Y., Bengio, Y., Bottou, L., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* **86** , 2278-2324.

Lee, C.-S., Wang, M.-H., Yen, S.-J., Wei, T.-H., Wu, I.-C., Chou, P.-C., Yan, T.-H. (2016). Human vs. Computer Go: Review and Prospect. *IEEE Computational Intelligence Magazine* **11**, 67-72.

Lindeberg, T. (1994). Scale-space theory: A Basic Tool for Analysing Structures at Different Scales. *Journal of Applied Statistics*, 225-270.

Liu, S., & Deng, W. (2015). Very Deep Convolutional Neural Network Based Image Classification Using Small Training Sample Size. *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 730-734. Kuala Lumpur, Malaysia.

Maalej, R., & Kherallah, M. (2020). Improving the DBLSTM for On-line Arabic Handwriting Recognition. *Multimedia Tools and Applications*, 17969-17990.

Mandelbaum, A. (2004). *The Metamorphoses of Ovid.*

McCarthy, J., & Feigenbaum, E. (1990). In Memoriam: Arthur Samuel: Pioneer in Machine Learning. *AI Magazine*, 10-11.

McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *The bulletin of mathematical biophysics*, 115-133.

Menabrea, L. F. (1842). *Sketch of The Analytical Engine Invented by Charles Babbage.* Bibliothèque Universelle de Genève.

Mostafa, S. S., Mendonça, F., García, A. R., Julia-Serda, G., & Dias, M. (2020). Multi-Objective Hyperparameter Optimization of Convolutional Neural Network for Obstructive Sleep Apnea Detection. *IEEE Access* **8**, 129586-129599.

Mualla, R., & Alkheir, J. (2018). Development of an Arabic Image Description System. *International Journal of Computer Science Trends and Technology*, 205-213.

Nongjiayuan, M. (2017, May 17). https://www.codenong.com/cs106165826/ .

Nwankpa, C., Gachagan, A., Ijomah, W., & Marshall, S. (2020). Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. *2nd International Conference on Computational Sciences and Technology, (INCCST).* Jamshoro.

O'Shea, K. T., & Nash , R. (2015). An Introduction to Convolutional Neural Networks. *ArXiv e-prints*.

Padmanabhan, A. (2019). Devopedia: https://devopedia.org/imagenet .

Plemakova, V. (2018). Vehicle Detection Based on Convolutional Neural Networks. *Master Thesis*. Tartu.

Rajesh, R., Kumaraswamy, R., & Kunhoth, S. (2011). Coherence Vector of Oriented Gradients for Traffic Sign Recognition Using Neural Networks. *Proceedings of the International Joint Conference on Neural Networks*, 907-910.

Rimol, M. (2019). Gartner: https://www.gartner.com/en/newsroom/press-releases/2019-11-14-gartner-forecasts-more-than-740000-autonomous-ready-vehicles-to-be-added-to-global-market-in-2023.

Roberts, L. (1963). Machine Perception of Three-Dimensional Solids. Cambridge, USA.

Rosenblatt, F. (1957). *The Perceptron: A Perceiving and Recognizing Automaton.* New York: Cornell Aeronautical Laboratory.

Rubik's code. (2018). https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 533-536.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 85-117.

Sebe, N., Cohen, I., Garg, A., & Huang, T. (2005). *Machine Learning in Computer Vision.* Springer .

Serna, C. G., & Ruichek, Y. (2018). Classification of Traffic Signs: The European Dataset. *IEEE Access*, 78136-78148.

Sharma, S. (2017). *Towards Data Science*. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

Sharma, S., Sharma , S., & Athaiya, A. (2020). Activation Functions in Neural Networks. *International Journal of Engineering Applied Sciences and Technology*, 310-316.

Shin, H.-C., Roth, H., Gao, M., Lu, L., Xu, Z., Nogues, I., . . . Summers, R. (2016). Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures,Dataset Characteristics And Transfer Learning. *Transactions On Medical Imaging*.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations.*

Stallkamp, J., Schlipsing, M., & Salmen , J. (2012). Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition.

Stenroos, O. (2017). Object Detection From Images Using Convolutional Neural Networks. Espoo, Finland.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A. (2015). Going Deeper with Convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9. Boston.

Szeliski, R. (2010). Computer Vision Algorithms and Applications. Washington, USA: Springer.

Şeker, A., Diri, B., & Balık, H. H. (2017). Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme. *Gazi Journal of Engineering Science* , 47-64.

Tomayko, J. (2003). Behind Deep Blue: Building the Computer that Defeated the World Chess Champion. *Technology and Culture*, 634-635.

Turing testi. (2021). Wikipedia: https://tr.wikipedia.org/wiki/Turing_testi

Turing, A. (1950). Computing Machinery and Intelligence. *Mind*, 433-460.

Wali, S. B., Hannan, M., Hussain, A., & Samad, S. (2015). Comparative Survey on Traffic Sign Detection and Recognition: a Review.

Walther, D., Riesenhuber, M., Poggio, T., & Itti, L. (2002). Attentional Selection for Object Recognition — A Gentle Way. *Proceedings of the Second International Workshop on Biologically Motivated Computer Vision* 472-479. Springer.

Wang, L., Shi, J., Song, G., & Shen, I.-f. (2007). Object Detection Combining Recognition and Segmentation. *Computer Vision-ACCV 2007*, 189-199. Tokyo.

Wang, S. C. (2003). Artificial Neural Network. *Interdisciplinary Computing in Java Programming*, 81-100.

Watkins, C. (1989, January). Learning From Delayed Rewards. *Ph.D. Thesis*.

Wikipedia. (2021). https://en.wikipedia.org/wiki/Library_(computing)

Willems, K. (2019). *Datacamp.* Datacamp: https://www.datacamp.com/community/tutorials/deep-learning-python.

Yamashita, R., Nishio, M., Do, R. K., & Togashi, K. (2018). Convolutional Neural Networks: An Overview and Application in Radiology. *Insights into Imaging*, 611-629.

Zeiler, M., & Fergus, R. (2013). Visualizing and Understanding Convolutional Neural Networks. *European Conference on Computer Vision*, 818-833.

Zheng, Y., Yang, C., & Merkulov, A. (2018). Breast Cancer Screening Using Convolutional Neural Network and Follow-up Digital Mammography. *Computational Imaging III.*

Zulkifli, H. (2018). *Towards Data Science*. https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

APPENDIX

Table A.1 Collective configuration settings of design parameters

	Learning Rate	Epoch	Batch Size	Filter Numbers	Activation Function	Input Image Size	Noise	Dro pout	Class Number	Train Image No / Test Image No
Lenet-5	0.01	15	32	6,16,120,84,43	Tanh	32x32x3	Not Added	No	43	39209/12630
Alexnet	0.01	30	32	96,256,384,384,256,4096,4096,43	Relu	227x227x3	Not Added	0.2	43	39209/12630
DP1	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP2	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	**Gaussian Noise(0.2)**	No	43	39209/12630
DP3	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	**Gaussian Noise(0.8)**	No	43	39209/12630
DP4	0.01	15	32	8,16,16,32,32,128,64,43	Relu	**32x32x1**	Not Added	No	43	39209/12630
DP5	0.01	15	32	8,16,128,64,43	Relu	**10x10x3**	Not Added	No	43	39209/12630
DP6	0.01	15	32	8,16,16,128,64,43	Relu	**64x64x3**	Not Added	No	43	39209/12630
DP7	0.01	15	32	8,16,16,32,32,128,64,43	**Tanh**	32x32x3	Not Added	No	43	39209/12630
DP8	0.01	15	32	8,16,16,32,32,128,64,43	**Sigmoid**	32x32x3	Not Added	No	43	39209/12630
DP9	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	**29344**/12630
DP10	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	**19520**/12630
DP11	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	**9664**/12630
DP12	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	**44**	**39500/12680**
DP13	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	**0.2**	43	39209/12630
DP14	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	**0.4**	43	39209/12630
DP15	0.01	15	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	**0.8**	43	39209/12630
DP16	**1**	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630

DP17	**0.5**	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP18	**0.1**	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP19	**0.05**	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP20	**0.01**	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP21	**0.001**	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP22	0.01	**5**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP23	0.01	**30**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP24	0.01	**60**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP25	0.01	**120**	32	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP26	0.01	15	**2**	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP27	0.01	15	**8**	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP28	0.01	15	**128**	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP29	0.01	15	**256**	8,16,16,32,32,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP30	0.01	15	32	**4,8,8,16,16**,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP31	0.01	15	32	**64,32,32,128,128**,128,64,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP32	0.01	15	32	8,16,16,32,32,**16,8**,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP33	0.01	15	32	8,16,16,32,32,**1024,512**,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP34	0.001	15	32	64,64,64,128,128,1024,1024,43	Relu	32x32x3	Not Added	No	43	39209/12630
DP35	0.01	15	128	64,128,128,1024,1024,43	Relu	64x64x3	Not Added	0.2	43	39209/12630

Table A.2 Collective performance results

	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Precision	Recall	F-Score	Time
Lenet-5	0.3539	0.8888	0.3759	0.8816	0.9135	0.8582	0.8846	9 mins
Alexnet	0.0952	0.9698	0.2209	0.9440	0.9552	0.9370	0.9459	30 hours*
DP1	0.2701	0.9132	0.2442	0.9279	0.9459	0.9158	0.9304	10.75 mins
DP2	0.3632	0.8831	0.2577	0.9188	0.9386	0.9047	0.9211	9.25 mins
DP3	0.5220	0.8286	0.3202	0.8991	0.9335	0.8725	0.9015	9.25 mins
DP4	0.4242	0.8691	0.7172	0.7861	0.8431	0.7538	0.7953	7 mins
DP5	0.9731	0.6802	1.5465	0.5626	0.6820	0.4995	0.5751	4.8 mins
DP6	0.0941	0.9711	0.1281	0.9632	0.9737	0.9580	0.9656	10 hours*
DP7	0.4041	0.8721	0.7689	0.7830	0.8238	0.7551	0.7874	9 mins
DP8	0.2173	0.9323	0.2663	0.9233	0.9415	0.9130	0.9268	9 mins
DP9	0.2641	0.9158	0.2668	0.9244	0.9409	0.9144	0.9273	8.3 mins
DP10	0.3222	0.8990	0.3306	0.8972	0.9271	0.8813	0.9033	5 mins
DP11	0.2362	0.9254	0.5233	0.8458	0.8745	0.8269	0.8496	3.25 mins
DP12	0.2824	0.9104	0.2459	0.9232	0.9413	0.9103	0.9253	9.75 mins
DP13	0.3163	0.8978	0.3249	0.9030	0.9250	0.8922	0.9081	9 mins
DP14	0.5544	0.8265	0.3792	0.8788	0.9201	0.8558	0.8861	9 mins
DP15	0.6992	0.7775	0.3900	0.8673	0.9191	0.8317	0.8724	9 mins
DP16	0.4810	0.8449	2703	0.8897	0.9173	0.8686	0.8919	36.5 mins
DP17	0.2689	0.9124	1981	0.8936	0.9112	0.8851	0.8977	40.5 mins
DP18	0.2120	0.9325	0.2152	0.9349	0.9504	0.9264	0.9380	48.5 mins
DP19	0.1705	0.9444	0.1874	0.9434	0.9558	0.9385	0.9470	54.5 mins

DP20	0.1384	0.9553	0.1826	0.9481	0.9549	0.9429	0.9488	58.5 mins
DP21	0.1369	0.9557	0.1805	0.9500	0.9584	0.9465	0.9523	72.5 mins
DP22	0.2767	0.9117	0.4676	0.8663	0.9012	0.8472	0.8729	6.25 mins
DP23	0.2336	0.9246	0.2064	0.9352	0.9520	0.9275	0.9394	18 mins
DP24	0.1857	0.9405	0.1950	0.9417	0.9533	0.9349	0.9439	27 mins
DP25	0.1203	0.9616	0.1473	0.9554	0.9645	0.9497	0.9569	36 mins
DP26	3.3061	0.1170	4.0423	0.1316	0.0263	0.0136	0.0178	22 mins
DP27	0.8167	0.7446	0.4890	0.8504	0.9201	0.8014	0.8526	13.5 mins
DP28	0.1400	0.9553	0.2406	0.9347	0.9451	0.9280	0.9364	10.75 mins
DP29	0.1803	0.9421	0.2030	0.9397	0.9533	0.9318	0.9423	9.75mins
DP30	3.4861	0.0574	17688	0.0285	0.0285	0.0285	0.0285	9 mins
DP31	0.0578	0.9818	0.1952	0.9470	0.9581	0.9420	0.9499	17 mins
DP32	0.4073	0.8725	0.3205	0.9028	0.9376	0.8817	0.9084	9 mins
DP33	0.2113	0.9312	0.2374	0.9306	0.9499	0.9216	0.9553	9.75 mins
DP34	0.0653	0.9801	0.1395	0.9624	0.9705	0.9553	0.9627	20 mins
DP35	0.0037	0.9988	0.0879	0.9751	0.9791	0.9734	0.9762	10.2 hours*

CV

Bünyamin DİKİCİ

Education Information

2017-	**Postgraduate** Gaziantep University Electrical and Electronics Engineering Department of Circuits and Systems
2012-2016	**Undergraduate** Osmaniye Korkut Ata University Electrical and Electronics Engineering

Job Experience

2020-	**Research Assistant** Gaziantep İslam Bilim ve Teknoloji University
2017-2020	**IT System Manager** Gaziantep University

Scientific Publication

13.12.2020	**ICOLES 2020** Serkan ÖZBAY- Bünyamin DİKİCİ Analysis of Convolutional Neural Network Design Parameters for Object Detection

Research Area

Artificial Intelligence, Machine Learning, Deep Learning, Optimization