REPUBLIC OF TURKEY

ADANA ALPARSLAN TÜRKEŞ SCIENCE AND TECHNOLOGY UNIVERSITY

INSTITUE OF GRADUATE SCHOOL

MEDICAL IMAGE REASONING WITH THE CONVOLUTIONAL NEURAL NETWORK - BASED FUZZY LOGIC

ESE AK

MASTER OF SCIENCE

ADANA 2022

ABSTRACT

MEDICAL IMAGE REASONING WITH THE CONVOLUTIONAL NEURAL NETWORK - BASED FUZZY LOGIC

Ese AK

Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Önder TUTSOY

September 2022, 116 pages

The new type of coronavirus (Covid-19), first detected in a local wild animal market in China's Hubei province in December 2019, has turned into a pandemic that has affected the whole world in a year. A definitive treatment for the disease has not yet been found. Although the vaccine development and applications have yielded positive results, the epidemic has not yet been brought under control worldwide. The most effective method in stopping the epidemic is the rapid detection of cases and the implementation of the quarantine measures together with the vaccination. The use of Convolutional Neural Netwoks(CNN), which has successful applications in the field of Computer Vison(CV) in recent years, as an auxiliary method for radiologists in the diagnosis of pandemic cases is the subject of this thesis. Within the scope of the study, the Covid-19 diagnostic methods are examined. The structure of the CNN is examined and its usage in image classification is explained. Different CNN architectures and development environments are introduced. Biomedical imaging techniques used in the diagnosis of the pandemics are briefly introduced. Preprocessing and data augmentation methods for image sets to be used in the CNN applications are discussed. The generated CNN model was trained and tested with Computed Tomohraphy(CT) chest images obtained from open sources. Test results were compared with studies in the literature. Fuzzy logic was applied to interpret the classification success of a specific sample with the trained network.

Keywords: Biomedical imaging, convolutional neural networks, Covid-19, fuzzy logic, pandemic diagnosis

ÖZET

BULANIK MANTIK TEMELLİ EVRİŞİMLİ SİNİR AĞLARI İLE MEDİKAL GÖRÜNTÜ YORUMLAMA

Ese AK
Elektrik Elektronik Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Önder TUTSOY

Eylül 2022, 116 sayfa

İlk olarak Aralık 2019'da Çin'in Hubei eyaletinde yerel bir vahşi hayvan pazarında tespit edilen yeni tip koronavirüs (Covid-19), geçen bir yıllık sürede tüm dünyayı etkisi altına alan bir pandemiye dönüşmüştür. Hastalığın kesin bir tedavisi henüz bulunmamaktadır. Aşı geliştirme ve uygulamaları olumlu sonuçlar verse de, salgın henüz dünya genelinde kontrol altına alınamamıştır. Salgının durdurulmasında en etkili yöntem aşılamayla birlikte vakaların hızlı tespiti ve karantina tedbirlerinin uygulanmasıdır. Son yıllarda bilgisayarlı görü(CV) alanında başarılı sonuçlar veren Evrişimli Sinir Ağları(CNN), pandemik vakaların teşhisinde radyologlara yardımcı bir yöntem olarak kullanımı bu tezin konusudur. Çalışma kapsamında, Covid-19 teşhis yöntemleri incelenmiştir. CNN'in yapısı incelenmiş ve görüntü sınıflama alanında kullanımı açıklanmıştır. Farklı CNN mimarileri ve geliştirme ortamları tanıtılmıştır. Pandemi teşhisinde kullanılan biyomedikal görüntüleme teknikleri kısaca tanıtılmıştır. CNN uygulamalarında kullanılacak görüntü setleri için ön işleme ve veri artırma yöntemleri irdelenmiştir. Oluşturulan CNN modeli açık kaynalardan elde edilen göğüs bilgisayarlı tomografi(BT) görüntüleriyle eğitilmiş ve test edilmiştir. Test sonuçları literatürdeki çalışmalarla karşılaştırılmıştır. Eğitilmiş ağ ile belirli bir örneğin sınıflandırma başarısını yorumlamada bulanık mantık uygulanmıştır.

Anahtar Kelimeler: Biomedical imaging, convolutional neural networks, Covid-19, fuzzy logic, pandemic diagnosis

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

ADAM	: Adaptive Moment Estimation Optimizer
ANN	: Artificial Neural Networks
CIFAR	: Canadian Institute for Advanced Research
Covid-19	: SARS-COV-2
CPU	: Central Processing Unit
CT	: Computed Tomography
CUDA	: Compute Unified Device Architecture
CV	: Computer Vision
GAN	: Generative Adversarial Networks
GGO	: Ground Glass Opacity
GPU	: Graphics Processing Unit
GRU	: Gated Recurrent Unit
ILSVRC	: ImageNet Large Scale Visual Recognition Challenge
k-NN	: k-Nearest Neighbors
LSTM	: Long Short-Term Memory
MNIST	: Modified National Institute of Standards and Technology
MODE	: Multi-Objective Differential Evolution
MRI	: Magnetic Resonance Imaging
NMR	: Nuclear Magnetic Resonance
NN	: Neural networks
ReLU	: Rectified Linera Unit
RT-PCR	: Reverse-Transcription Polymerase Chain Reaction
RMSprop	: Root Mean Square Propagation
SARS-COV-2	: Severe Acute Respiratory Syndrome causing Coronavirus-2
SVM	: Support Vector Machines

1. INTRODUCTION

The SARS-COV-2 virus, briefly called the Covid-19, turned into a pandemic that spread from China to the whole world in 2019(Shi et al., 2020) (Hamed, 2020). Although the Covid-19 vaccine development and application studies continue, the epidemic has not yet been brought under control. The most effective way of prevention for this disease, for which there is no definite treatment method, is to prevent the spread of the disease among people. For this reason, rapid and accurate diagnosis of the disease is of vital importance in preventing the disease. Although the commonly used method in the diagnosis of the Covid-19 is Reverse-Transcription Polymerase Chain Reaction(R-PCR) tests(Vinh et al., 2020), studies have shown that biomedical chest images (MRI, CT, X-Ray) are valid methods in the diagnosis of the Covid-19(Chung et al., 2020) (Ai et al., 2020) (Bai et al., 2020) (Huang et al., 2020) (Liu et al., 2020) (Simpson et al., 2020). Diagnosis of the Covid-19 from chest images requires expertise and effort in radiology, as the Covid-19 shows similar symptoms in lung tissue as other pandemics(Bai et al., 2020). Studies have shown that the use of deep CNN in the diagnosis of The Covid-19 from chest images is a helpful method for radiologists in making the diagnosis faster and more accurate(Bai et al., 2020) (Yasar & Ceylan, 2021) (Ates et al., 2020) (Narin et al., 2020) (Ismael & Şengür, 2020) (Apostolopoulos & Mpesiana, 2020) (Dansana et al., 2020) (Zhang et al., 2020) (Bressem et al., 2020) (Tilborghs et al., 2020) (Yan et al., 2020) (J. Ma et al., 2021) (Pham, 2020) (Marques et al., 2020) (Singh et al., 2020). Within the scope of this thesis, the diagnosis of the Covid-19 from the CT images was examined with the CNN application.

This thesis consists of six main chapters parts. The first chapter is a summary for the other chapters. The second part summarizes the sources referenced in the thesis. The Materials and Methods section, which is the main theme of the thesis, consists of six titles. In the first chapter, ANN, which is the basis of CNN, is examined. In the second part, CNN architecture is explained. The third subsection is a summary presentation about fuzzy logic. The fourth chapter of third main section deals with data preparation for CNN applications in four subtitles. These topics are biomedical imaging techniques, the Covid-19 datasets for CNN, data preprocessing and data augmentation, respectively. In the fifth chapter of section three, frameworks that provide convenience for coding CNN are introduced. In the sixth subchapter,

successful classical CNN architectures are examined. In the seventhth subchapter of section three, the realization of CNN with the selected framework(Matconvnet) structure is covered. The created CNN structure was trained and tested with images compiled from open sources in the fourth main section. The obtained results were compared with the studies in the literature. Applying fuzzy logic to interpret the classification success of a specific sample with the trained network, with the overall statistical success of the network, is the last subsection of this chapter. The conclusions reached in the thesis are presented in the fifth chapter. The recommendations made according to the conclusions constitute the sixth section.

1.1. Traditional Methods for the Covid-19 Diagnosis

Although the widely used method in the diagnosis of Covid-19 is RT-PCR tests, clinical findings and radiological images are also methods that contribute to the diagnosis of the disease(Vinh et al., 2020) (Ai et al., 2020). Although the symptoms of the Covid-19 in lung the CT images are similar to other viral diseases, experienced radiologists have shown successful results in the diagnosis of the Covid-19 from chest the CT(Vinh et al., 2020) (Bai et al., 2020). The Covid-19 findings on chest the CT images are areas of consolidation and GGO with bilateral peripheral involvement in multiple lobes that progress towards "crazy laying" patterns and consolidation(Chung et al., 2020) (Ai et al., 2020) (Bai et al., 2020) (Huang et al., 2020) (Liu et al., 2020) (Yasar & Ceylan, 2021). Although not as widely used as the CT and X-ray images(Narin et al., 2020) (Ismael & Şengür, 2020) (Apostolopoulos & Mpesiana, 2020) (Dansana et al., 2020), MR images(Ates et al., 2020) can also be used in the diagnosis of the Covid-19. There are various studies reveal the success of using CNN in the diagnosis of the Covid-19 from the CT images. Different modifications of the CNN architecture have been applied in these studies(Zhang et al., 2020) (Bressem et al., 2020) (Pham, 2020) (Marques et al., 2020) (Singh et al., 2020). Some of studies are about the diagnosis of the Covid-19. Diagnosis is in the form of binary or multiple classifications. Another part is related to the segmentation of the Covid-19 symptoms in lung the CT image(Tilborghs et al., 2020) (Yan et al., 2020) (J. Ma et al., 2021).

1.2. Convolutional Neural Networks

ANN is a mathematical computational model inspired by the biological nervous system(O'Shea & Nash, 2015). The purpose of this model is to find approximate non-parametric solutions to problems that are difficult to solve with the mathematical

deterministic methods. In this model, it is aimed at reaching the best solution rather than the exact solution. The optimization process of the ANN is essentially based on the black box approach that learns by updating its unknown parameters based on the input data and the estimated model output(O' Mahony et al., n.d.). The learning information of the model is stored in the connection weights known as the unknown ANN parameters between the neurons that make up this black box.

The CNN, which has common applications in computer vision, is a specialized model of ANN(X. Ma et al., 2017) (Li et al., 2020). It is preferred because it reduces the number of trainable parameters and automates the feature extraction processes(Yamashita et al., 2018) (Jin et al., 2015) (Yamashita et al., 2018). (Wang et al., 2018) The source of these abilities is the additional layers in its structure. These layers are convolution layer, pooling layer, fully connected layer and classification layer(Narin et al., 2020) (Yamashita et al., 2018). In addition, the special transfer functions applied at the output of these layers are also the components of this network(Murphy, 2016). The number of channels in the input data has added the concept of depth to these networks(O'Shea & Nash, 2015) (Wu, 2017). This network structure is called Deep CNN. The depth concept is also related to layer count in CNN(O'Shea & Nash, 2015) (Namatēvs, 2017).

1.3. Biomedical Imaging Techniques

Monitoring the structures of organs and tissues in the human body are methods that are widely used in the diagnosis and treatment of diseases. Medical imaging techniques, which started in the 1890s, showed a gradual development after the 1980s(Ilangovan et al., 2016). Imaging techniques, which have developed rapidly in recent years, are based on different physical, chemical and nuclear phenomena. X-ray and the CT imaging, which is the subject of this thesis, is based on X-rays(Sun et al., 2012). Another imaging technique discussed in this thesis, MRI, is based on the detection of electromagnetic radiation emitted by atomic particles as a result of spin motion in a magnetic field(Möllenhoff et al., 2012). Apart from the scope of this thesis, nuclear isotope-based, ultrasonic wave-based and hybrid techniques are also commonly used imaging techniques in the field of medicine(Ilangovan et al., 2016).

1.4. Biyomedical Image Datasets

The training dataset is as important as the network architecture in the success of deep learning algorithms. The dataset should contain a broad spectrum which can generalize the different patterns that may be encountered in the tests. It should also be concise enough to guarantee completion of training in a reasonable time. There are various sets of the Covid-19 CT (Soares et al., 2020) (Yang et al., 2020) and X-ray(Irvin et al., 2019) (Shuja et al., 2021) images compiled from open sources on the Internet. While some of these data are presented in raw form, some are preprocessed for compatibility with deep learning algorithms. Some of these image sets are arranged for binary classification (Soares et al., 2020) (Yang et al., 2020). Others are data suitable for multiple classifications(Irvin et al., 2019) (Shuja et al., 2021), including different findings in addition to the Covid-19. In a comprehensive survey of the COVID-19 open source data, the data are classified as visual, textual, and audio data. Visual data were analyzed as the CT and X-ray(Shuja et al., 2021).

1.5. Image Prepocessing Techniques

The CNN can automate the process of extracting features from the images but input images must meet the certain standards for accurate diagnosis. Therefore, images obtained from the different sources need to be pre-processed before they are applied to the network(Basheer, I.A., 200 C.E.) (Jeong et al., 2018). For CNN, the input images must be equally sized. Because the number of parameters in the convolution and pooling layers depends on the size of the input images. It is suggested that the input image dimensions should be sequentially divisible by two so that the data sizes at the output of the layers are integers(O'Shea & Nash, 2015). In addition, the brightness level should be appropriate in order to recognize the image features. It is also important that all the images are given to network in same number format. Because, giving an image to network defined between 0 -1 (float) and an image defined between 0-255(integer) together will cause the float values to become meaningless. Textural and edge enhancements in the image also affect the success of CNN. For compatibility with the CNN model, the color channel numbers of the input data must be equal. Resizing, mean normalization, histogram adjustment, and standardization are common image preprocessing methods in the CNN applications(Sudeep & Pal, 2016) (Bow, 2002).

1.6. Training and Testing of the CNN

In this thesis, CNN model was applied to diagnose the Covid-19 from the CT images. To create the CNN algorithm the MatConvNet library in Matlab R2018B is considered. Two

different data sets were used in training and testing applications. One of them is SARS-CoV-2 CT-scan Dataset(Soares et al., 2020) was used as training validation and test data. 1024 images from the Covid-19 positive and the Covid-19 negative classes were used in the training of the network. Of the remaining images, 100 images for each class were reserved for validation and test sets. The other data set was prepared by (Yang et al., 2020). In the second dataset, 180 images were used for training and 60 images were used for validation and testing. In experimental studies, these data sets have been increased 20 times by using rotation and translation transformations. The created CNN models were trained with different training parameters and the results were interpreted.

1.7. Comparative Resuslts

There are a number of parameters that affect the success of deep learning algorithms. These are the structural parameters of the model, the training parameters of the model, the data set used in the training of the model, the preprocessing applied to the data set and the indirectly used hardware. Structural parameters are input image sizes, number of layers, number of filters, filter sizes, activation functions, etc. Training parameters are batch size, learning rate, dropout rate, etc. The parameters related to the data set are the number of data, the quality of the data, the method of separating the training and test data, the data augmentation methods, etc. Hardware parameters are the amount of memory, processor capacity, etc.

A total of fifty-five different applications were made with eight CNN models and two data sets used in this study. The best accuracy and F1-score values obtained are equal to 0.96. The success rate stated in the article in which the first data set is presented is 0.97, which is higher. The best result for the accuracy and F1-score obtained in the study with the second dataset was 0.68, lagging behind the 0.89 - 0.90 ratio in the reference article. Although the test results seem reasonable, this success could not be achieved in cross-checks between these two data sets. The best accuracy and F1-score values obtained in the cross-tests were 0.59 and 0.66, respectively.

A summary list of similar studies in the literature is presented in Table 4.10. When the table is examined, the existence of higher classification successes will be seen. It will be seen that the methods used in these studies are proven deep learning architectures and SVM method (Soares et al., 2020) (Marques et al., 2020).

Considering the other studies in the table, it can be said that the results of this study are reasonable. However, the reason for the low success rate in cross-testing with two different datasets may be the subject of a future study(Table 4.7).

2. LITERATURE REVIEW

Since this thesis includes different subjects, a wide literature list was created within the scope of the research. In order to present the literature list more systematically, the review is grouped under four headings. The first part includes studies on the Covid-19 pandemic and diagnostic methods. The second part is about CNN and its based ANN. The third section presents studies on CNN applications in the Covid-19 diagnosis and last section, about other references related to the thesis are listed.

2.1. Covid-19

Since this thesis focuses on the diagnosis of the Covid-19 from biomedical images, the literature is limited to this area. Diagnosis of the disease by clinical methods is beyond the scope of this study. In a detailed study, the history of the epidemic, ways of transmission and spread, mortality rates are given in comparison with other coronavirus pandemics. The lesions of the disease on chest CT are described(Shi et al., 2020). In the study (Hamed, 2020), the origin of the name of the coronavirus, epidemic status, clinical findings, drug development and immune response, genetic diversity, drug treatment potential, possible treatment concepts, and control issues were examined. The R-PCR test and chest the CT image in the diagnosis of the Covid-19 were examined(Vinh et al., 2020).

In Study (Chung et al., 2020), the Covid-19 symptoms on chest the CT images are described in detail. Study (Ai et al., 2020) examined the correlation between the R-PCR test and the results of chest the CT images in diagnosing the Covid-19. In the study (Huang et al., 2020), the clinical findings of 41 patients who were diagnosed with the Covid-19 in Wuhan, China were examined. In study (Liu et al., 2020), the etiology and epidemiology of the Covid-19 pandemic was examined. Clinical findings of the disease are presented. Abnormality in chest the CT images is presented comparatively for the Covid-19, SARS and MERS. Study (Simpson et al., 2020) provides a radiological guideline for chest the CT.

2.2. Background Researches in the CNN

It is not possible to know CNN without understanding ANN. Because traditional feed-forward ANN forms both the foundations and the struct of CNN. For this reason, it would be appropriate to sequentially examine the wide literature in this field.

Study (Keskenler & Keskenler, 2017) provides a historical overview of ANN. Study (Li et al., 2020) referred to the chronological development of ANN. The transition from ANN to CNN is explained. In the study (Sadowski, 2017) provides a brief summary on backpropagation.

The study(Basheer, I.A., 200 C.E.) gives detailed information about, artificial neuron model. Explain the application areas, learning rules, back propagation algorithm, determination of training and test data, data preprocessing, data enrichment, data balancing, number of iterations, stopping criteria, batch and stochastic training model, number of hidden layers, parameter optimization. Study (Abraham, 2005) gives a comprehensive introduction to ANN. Study (Nur et al., 2014) provides an overview of weight update methods for ANN. Study (Abiodun et al., 2019) provides a detailed look at the use of ANN in pattern recognition. This study explains the use of CNN as a special type of ANN in pattern recognition. Study (Lederer, 2021) provides a detailed presentation of the activation functions commonly used in ANN applications. Study (Ding & Qian, 2018) gives a brief presentation of the activation functions commonly used in ANN applications.

Study (Attoh-Okine, 1999), examined the effects of learning rate and momentum term for backpropagation ANN. Study (Yam & Chow, 2000) provides a method for determining initial weights for the ANN. Study (Lehtokangas et al., 1995) provides an initial weight adjustment method for multilayer perceptron networks.

Study (Hinton et al., 2012) examined the issue of increasing NN success with dropout application. Study (Khan et al., 2018) proposes a new dropout method to strengthen the generalization property of deep NNs. Study (Inoue, 2020) proposes a new dropout method. After a brief summary about CNN, modern CNN architectures are explained in detail A more detailed study on CNN is (Wu, 2017). In this study, the mathematical foundations of CNN and equations for back propagation are given. Another detailed study gives hints on adjusting the structural parameters for a successful CNN model(O'Shea & Nash, 2015).

Study (Namatēvs, 2017) gives a brief description of CNN. The study indicates the application areas of CNN. It offers a detailed look at the historical development of CNN. The study (Albawi et al., 2018) highlights the aspect of CNN to reduce network training parameters and

extract abstract features. Study (Yamashita et al., 2018) provides explanation on CNN terminology, architecture, layers, training and application to different classification problems. Using of CNN in radiology is explained in detail according to fields. Study (Wang et al., 2018) provides a historical overview of ANN and CNN. The study (O' Mahony et al., n.d.) compares traditional and deep learning algorithms on CV. The study (Radiuk, 2018), investigated how batch size affects the performance of CNN. Study (Sudeep & Pal, 2016) investigated the effect of data processing methods on the classification success of CNN.

Study (Shorten & Khoshgoftaar, 2019), provides a detailed look at data augmentation techniques for Deep learning algorithms. The importance of data augmentation was emphasized in the study. Study (Inoue, 2018) introduces a new image data augmentation method for deep learning algorithms. Study (Bahrampour et al., 2015) provides a detailed report about deep learning framework softwares.

2.3. Diagnosis of the Covid-19 with CNN

The diagnosis of the Covid-19 with deep CNN, which is studied within the scope of this thesis, is an auxiliary method that provides speed and convenience for radiologists. A review of some studies examined in this context is below.

Study (Yasar & Ceylan, 2021) presents a comparative study of the diagnosis of the Covid-19 from the CT images. Study (Ates et al., 2020) presents a comparison of the CT chest images and MRI images in the diagnosis of the Covid-19. Five pre-trained CNN models were applied to diagnose the Covid-19 from X-ray images in Study (Narin et al., 2020). Study (Ismael & Şengür, 2020) presented a new deep learning model for diagnosing the Covid-19 from X-ray images. The aim of Study (Apostolopoulos & Mpesiana, 2020) is the Covid-19 diagnosis with state-of-the-art pre-trained CNN.

Study (Rahman et al., 2020) includes automatic diagnosis of bacterial and viral pneumonia from chest X-ray images with pre-trained CNN. The study (Dansana et al., 2020) focused on the Covid-19 diagnosis with deep learning algorithms from the CT and X-ray images. Study (Bressem et al., 2020) compares different deep learning algorithms in diagnosing the Covid-19 from chest X-ray images. Study (Tilborghs et al., 2020) compares the performance of deep learning algorithms in segmentation of chest the CT images of the Covid-19 patients to

identify lesion and lesion type. Study (Yan et al., 2020) presents a three-dimensional deep CNN model named COVID-SegNet built for the Covid-19 segmentation from the CT images. Study (J. Ma et al., 2021) focused on the Covid-19 segmentation with small size datasets.

Study (Pham, 2020) compared the performance of 16 pre-trained CNN models in diagnosing the Covid-19 from chest the CT images. Study (Marques et al., 2020) presents the CNN model study for the Covid-19 diagnosis using the EfficeNet architecture. Study (Singh et al., 2020) presents a new deep learning model for diagnosing the Covid-19 from the CT images. In the study, the CNN initial parameters were adjusted using the multi objective differential evolution. Successful machine learning algorithms were compared with the generated CNN.

The study (Soares et al., 2020) presented a publicly available SARS-CoV-2 CT scan dataset. The created 23-layer CNN is trained with this dataset. The same dataset is also used with familiar deep CNN architectures and machine learning methods. The results are presented comparatively. Study (Yang et al., 2020) presented a dataset compiled from 216 patients. 349 of the data were diagnosed as Covid-19(+), 416 of them were diagnosed as Covid-19(-). Study (Cohen et al., 2020) presented 679 chest X-ray images from 412 people from 26 countries. Study (Shuja et al., 2021) has classified the Covid-19 datasets presented in open sources. This classification is made according to data types and applications. Study (Irvin et al., 2019) presented a dataset of 224316 chest X-ray images obtained from 65240 patients.

2.4. Other Related Literatures

Study (Ilangovan et al., 2016) provides a brief overview of biomedical imaging techniques. Study (Sudeep & Pal, 2016), highlighted the success of normalization and standardization preprocessing methods for different CNN models. Study X is a detailed study that categorically classifies data augmentation methods for deep CNN.

Study (Sun et al., 2012) provides a brief overview of industrial three-dimensional the CT imaging techniques. Study (Moore et al., 2011) presents a computer model to generate realistic simulated computed radiography chest images using real patients the CT datasets. Study (Möllenhoff et al., 2012) provides a historical presentation of MRI. Study (Doneva, 2020) provides an overview of image reconstruction patterns in MRI imaging.

A detailed document about the Matconvnet library where the CNN models in the thesis were implemented was presented by (Vedaldi et al., 2015). The study (Mocanu, 2008) explained the required library structure for running deep learning networks on GPU. Detailed information on the development of fuzzy logic, fuzzy set theory, the structure of fuzzy systems, and application areas is presented in (Sivanandam et al., 2007). Study (Işıklı, 2007) is a descriptive document for the historical development of fuzzy logic theory and fuzzy logic theory.

3. MATERIALS AND METHODS

The ANN, the foundation of the CNN is a non-parametric computational model inspired by the biological nervous system (O'Shea & Nash, 2015). The key purpose of this model is to create a general solution that can be adapted to different numeric problems. This model is a predictive model that sets out the rule of complex relationships by testing a large number of simple events(O' Mahony et al., n.d.). The input and output values are measured mostly in binary, decimal, or real numbers. The general rules are coefficient that stored in neurons. These coefficients are adjusted by evaluating the output from the input during the training of the NN. Training ANN with this method is a supervised learning model. In some ANN applications, there is no expected output for input data. These networks are the type of network that reveals hidden relationships between input data(O'Shea & Nash, 2015). This type of ANN learns with unsupervised methods. CNN, the focus of this thesis, is mainly based on the traditional supervised ANN(Basheer, I.A., 200 C.E.). Traditional feedforward supervised ANN's learn with the feedback generated from output errors.

CNN promises to solve several CV problems with its additional layers. These layers are the convolution layer and pooling layer(Wu, 2017). In addition, the transfer functions applied at the output of these layers are also the components of this network. The fully connected layer and classification layer, two of CNN's layers, are similar to classic ANN. The number of channels in the input images has given these networks the concept of depth. This network structure, which is widely used in the field of CV, is called Deep CNN. The depth concept is also related with the number of layers in the fully connected layer(O'Shea & Nash, 2015).

3.1. The ANN Architecture

Before introducing the CNN, it would be helpful to provide an overview of the classical (referred to as "traditional" in some sources) ANN. The ANN is a computational model that emulates the obscurity of the human brain. Similarly, the basic element of both structures is the neuron. The artificial neuron is the mathematical equivalent of the biological neuron. In general terms, ANN is the equivalent of the entire biological nervous system. It basically consists of input layer, hidden layer and output layer.

3.1.1. Structrue of Artifical Neuron

Artificial neurons, like biological neurons, consist of three parts. These are inputs, body and output. There may be more than one entrance to the neuron body. The output of each neuron is equal to the sum of the products of that neuron's inputs and input weights.(Figure 3.1)

$$y_o = \sum_{i=1}^{n} w_i x_i + b \tag{1}$$

where x is the input data for the neuron and n is number of x. Usually, the input data is expressed as a vector because the number of input data is more than one. w is the weight vector that determines the contribution of each input to the neuron output. The learning information of the neuron is represented by this vector. The basis for training the ANN is to tune this vector. When the input data or weight is 0, the output is also 0. The bias value b is added to avoid a zero output value. Because the zero output value does not transfer any information to the next layer.

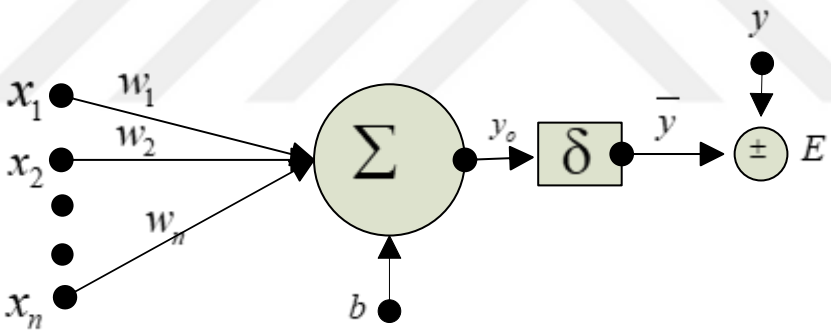

Figure 3.1. Artificial neuron model

A non-linear activation function δ is applied to the neuron output value to prevent excessive growth of the output value.

$$\bar{y} = \delta(y_o) \tag{2}$$

This function ensures that the output remains in a certain range for all real values produced by the neuron.

3.1.2. Backpropogation for a Single Neuron

In order to evaluate the learning level of the ANN, the error value E produced at the output is calculated. This value is usually derived from the difference between the expected value and the produced value.

$$E = y - \bar{y} \tag{3}$$

where y is the output value we expect the network to produce for each sample input. $\bar{y}$ is the value which is produced by the neuron and activation function, known as the estimated or learned output. To reduce this error value, a correction calculation is made from output to input. This is called back propagation. Backpropagation requires calculating the derivative of the error with respect to each input weight. This computation, which is simple for a single neuron, brings a high computational cost for a multilayer NN. To overcome this difficulty, the chain rule used to calculate the derivative of the compound functions is used.(Basheer, I.A., 200 C.E.) The real output value of NN is composed of all connected neurons output values before it. Backpropagation for e single neuron is calculated as follows. Derivative of the error with respect to the neuron output:

$$\frac{\partial E}{\partial \bar{y}} = -1 \tag{4}$$

Derivative of neuron output with respect to activation function:

$$\frac{\partial \bar{y}}{\partial y_o} = \delta(y_o).(1 - \delta(y_o)) \tag{5}$$

Derivative of the sum of the products according to the input weights:

$$\frac{\partial y_o}{\partial w_i} = X_i \tag{6}$$

When chain rule formula is applied,

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial \bar{y}} . \frac{\partial \bar{y}}{\partial y_o} . \frac{\partial y_o}{\partial w_i} \tag{7}$$

Substitute for piecewise derivatives:

$$\frac{\partial E}{\partial w_i} = (-1).X_i.\delta(y_o).(1-\delta(y_o)) \tag{8}$$

Finally, when the forward calculation values are written for the neuron:

$$\frac{\partial E}{\partial w_i} = (-1).X_i.\delta(Wi.Xi+b).(1-\delta(Wi.Xi+b)) \tag{9}$$

In multilayer networks, backward calculated derivatives are stored for each layer. These values are used in the derivative calculation of the previous layer.

3.1.3. Structrue of the Multi-Layer ANN

The ANN consists of a multi-layered structure with a different number of neurons in each. The first layer is the input layer where the data is received. The output of each neuron in the input layer is connected to the hidden layer. The last layer is the output layer of the network, and the performance of the network is evaluated according to the result obtained from this layer. Intermediate layers are fully connected (hidden) layers that store the learning weights(Figure 3.3). The hidden layer consists of multiple layers that are added sequentially. Each of these layers can have a different number of neurons. The output layer is fully connected to the last hidden layer. The number of neurons in the output layer usually depends on the task type of the ANN. (Figure 3.2)

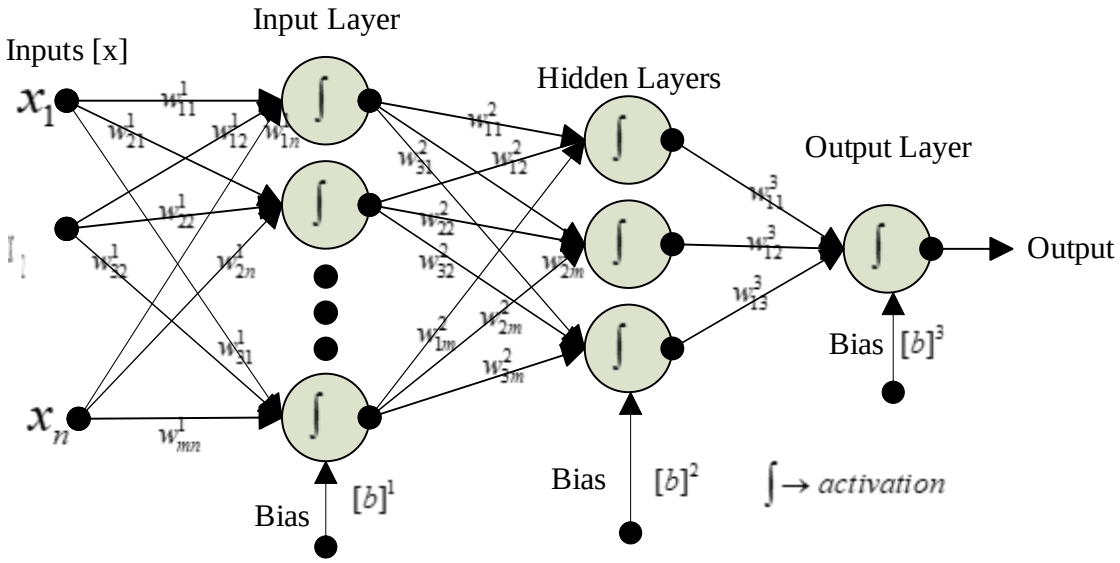

Figure 3.2. Structure of Conventional Neural Networks

where x and b are vectors and w is a two-dimensional matrix. The upper index of w indicates the number of the layer, and the lower indices respectively show the number of the neuron in this layer and the number of the neuron to which the related neuron is connected in the previous layer. After this demonstration, both forward and backward propagation calculations can be defined as matrix and vector operations.

3.1.4. Avoiding overfitting

When we think of ANN as a black box, this black box makes a mapping from input to output. When the combination number of paths in the black box is too big, a so-called learning can occur by matching a separate path for each entry.

This network can be very successful for training data, but fails to classify data it has not seen before. This is called overfitting. For such network layer, the comparison of a crowded staff and an inefficient company would be appropriate. In such a company, every employee memorizes a very simple job. When a new task is assigned, no suitable employee can be found. Employees will be more creative when some employees are dismissed in this company and their duties are distributed to other employees. In this case, costs will decrease and staff will be more prepared for new tasks. Some units in the hidden layer are removed to prevent overfitting. Instead of dropping the neuron completely, it is suggested to multiply the output value by 0.5 to make the mean value consistent with the training time (Inoue, 2020). This process is called dropout (Figure 3.4). It has been shown that dropping randomly 50% of neurons in each iteration prevents overfitting(Hinton et al., 2012).

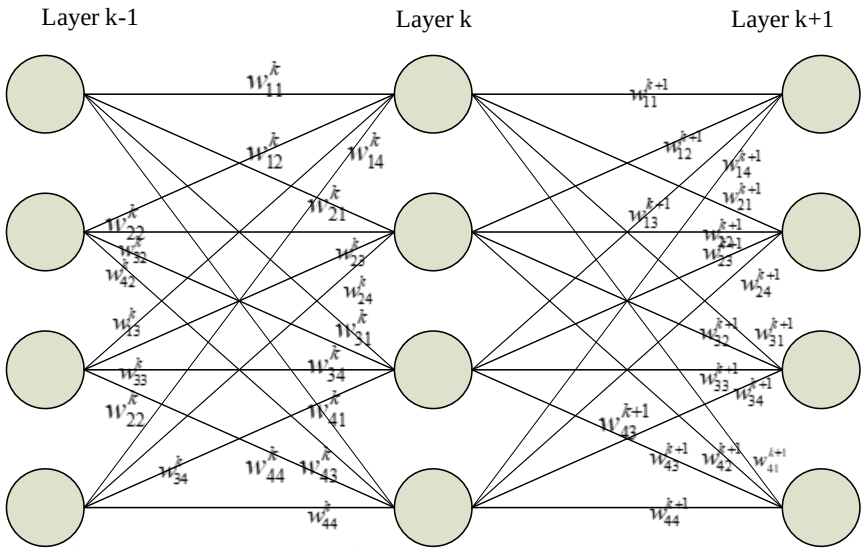

Figure 3.3. Fully connected layer.

Dropping connects, a different type of dropping, has been proposed (Figure 3.5). This method drops some of connections instead of the dropping neurons in the layer depending on the certain probability (Inoue, 2020) (Khan et al., 2018). The weights of these connections are distributed among other connections.

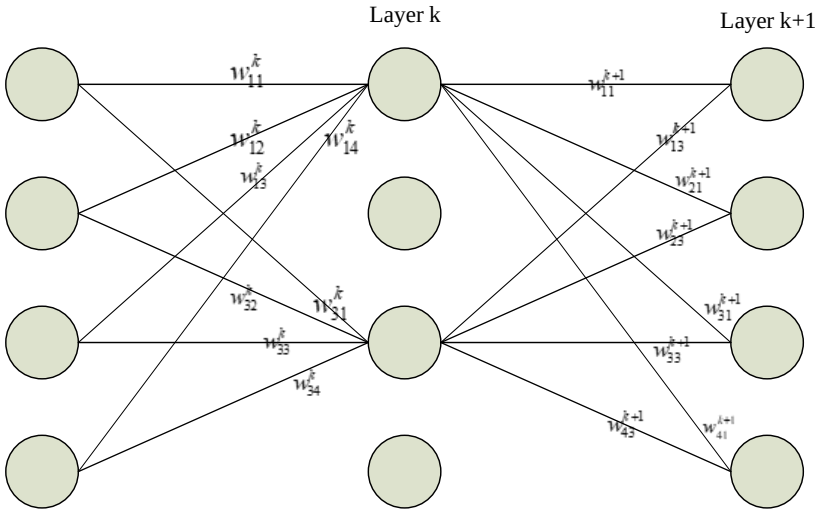

Figure 3.4. Dropout layer.

Thus, each neuron contributes to each output. A different type of dropout is multi-sample dropout proposed in (Inoue, 2020).

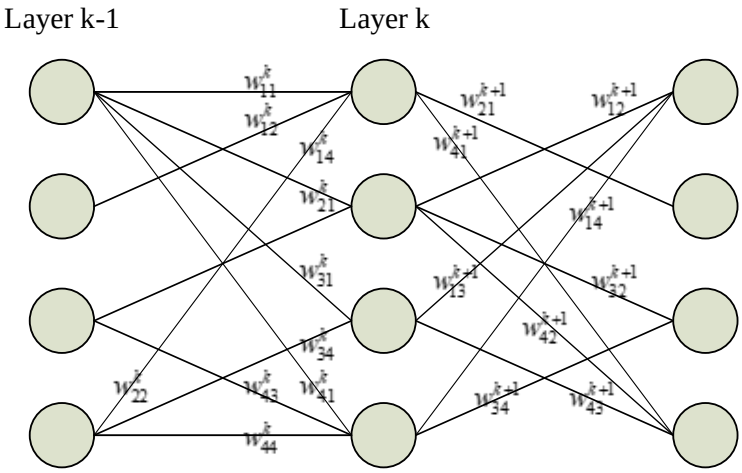

Figure 3.5. Drop connect layer

In this technique, multiple dropouts are applied in one layer and the average of their error values is used for backpropogation.

3.1.5. Setting the error correction rate

Moving in the opposite direction of the gradient vector leads to the minimum value point of the function relatively. In ANN, the minimum value of the error(objective) function provides the best performance. Supporting the neuron input weights with a negative gradient adjusts the weights to reduce the output error (Wu, 2017) (Figure 3.6).

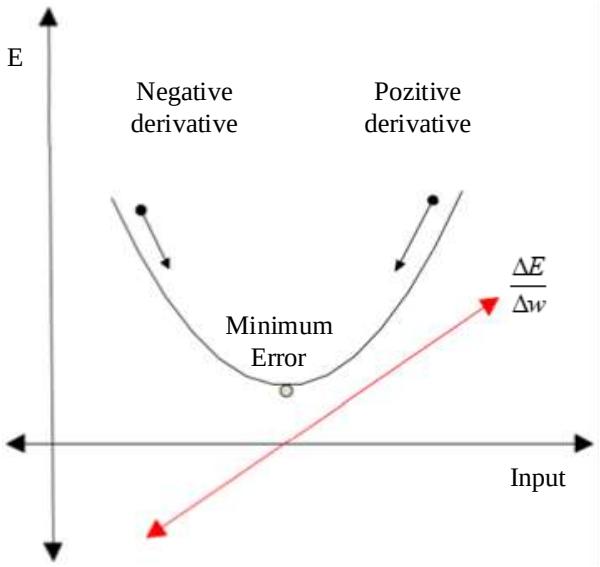

Figure 3.6. Moving in the anti derivative direction

The amount of change is important when adjusting weights with the gradient vector (Wu, 2017). An appropriate coefficient determining the effect of the gradient vector on weight provides the most acceptable learning performance (Figure 3.7)

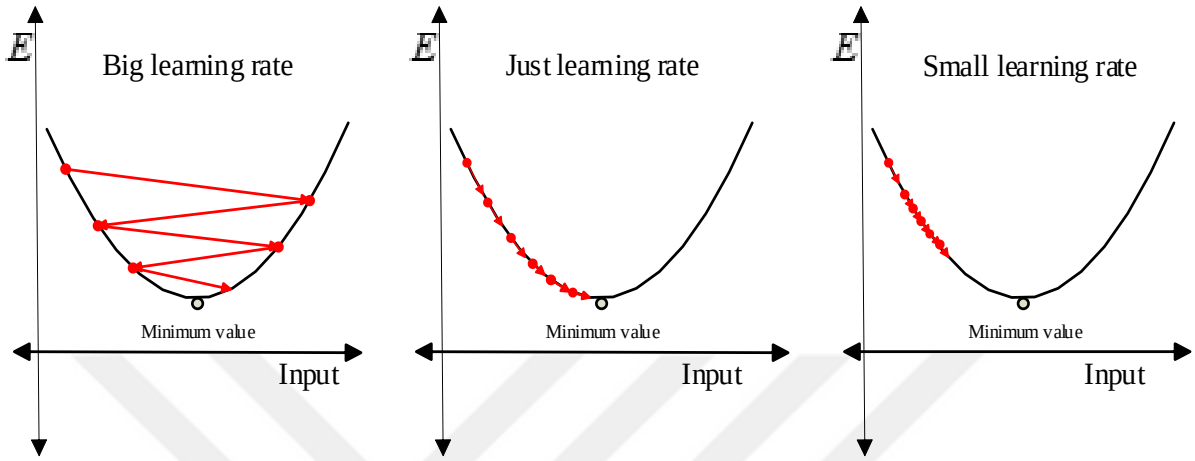

Figure 3.7. Effects of different learning rate

This coefficient μ is called the learning rate. If the learning rate is too large, it causes the optimum point to be missing out. It's very small size also increases the training time of the network. Getting the best learning rate is a major challenge (Basheer, I.A., 200 C.E.) (Attoh-Okine, 1999).

$$\Delta w_t = -\mu \frac{\partial E}{\partial w_t} \tag{10}$$

where Δw_t changing value of w_t. This training of ANN is called the Delta Rule.

$$w_{t+1} = w_t + \Delta w_t \tag{11}$$

Updating the weights for each training data causes significant differences in error values. In this method, data containing noise causes oscillation while appropriate training data provides fast convergence. The disadvantage of this method is that the training that starts with an unsuitable sample may force the search wrong direction. This method is called Stochastic Gradient Descent(Wu, 2017) (Murphy, 2016).

3.1.6. Different approaches for weight updating

Saving errors for a certain amount of training data and updating the weights according to their average is called Batch Gradient Descent. The number of this data group is called batch size. In this method, there is less oscillation in the amount of error during the training. Each loop requires less mathematical operations. However, the number of iterations required for sufficient training may increase. Other disadvantages of batch training are high memory requirements and the problem of stuck at the local minimum (Basheer, I.A., 200 C.E.; Wu, 2017) (Radiuk, 2018) (Figure 3.8). The solution to this problem is to reduce the batch size. This is called a mini batch. Batch size adds a new dimension to the input tensor. For example, when creating 32-image batch consisting of 64x64 color images, the input tensor will be 4-dimensional. These dimensions are 64x64x3x32.

It is important to transfer the historical gradient vectors to the new iteration to protect the gradient continuity from the noise of extreme data. The coefficient that transfers the gradient vector from previous iterations to the new iteration at a certain rate is called the momentum coefficient(Eq:12) (Basheer, I.A., 200 C.E.) (Murphy, 2016).

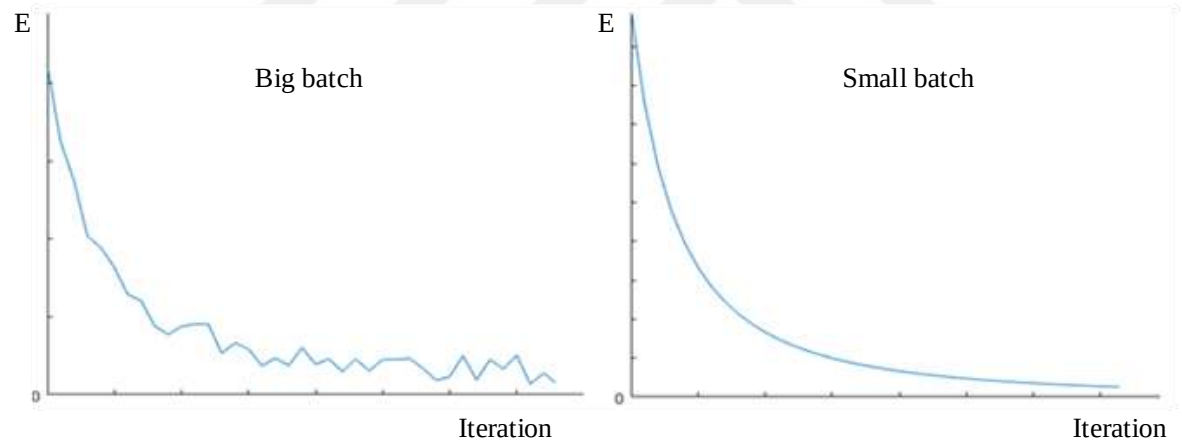

Figure 3.8. Effects of big and small batch size on convergence graph

$$\Delta w_i(n) = -\mu.\frac{\partial E}{\partial w_i} + \alpha.\Delta w_i(n-1) \tag{12}$$

where a is momentum coefficient and n is iteration number (Abraham, 2005). Determining momentum and batch size is a challenge in ANN architecture as well as learning rate(Radiuk, 2018).

3.1.7. Network Initial Values and Optimization Thecniques

Another challenge is to determine the initial value for the network weights. Initial vale of weight is critical for reliable converging amount and learning time. Although various methods are proposed in this area, there is no definite solution (Attoh-Okine, 1999) (Yam & Chow, 2000) (Lehtokangas et al., 1995). In today's deep learning networks, transfer learning or pre-trained networks can be shown as a solution to this problem (Narin et al., 2020) (Apostolopoulos & Mpesiana, 2020) (Rahman et al., 2020).

Depending on the starting weights, the error value reached may not always be the best value.(Figure 3.9) This point is called the local minimum. Getting rid of local minimums is another challenge for ANN. The weakness of the back propagation is the loss or increase of the gradient(Namatēvs, 2017). Reducing the number of hidden layers which is the solution for this problem, may causes insufficient learning. For example, the solution of the XOR problem for ANN has been possible with the discovery of multilayer networks (Wang et al., 2018).

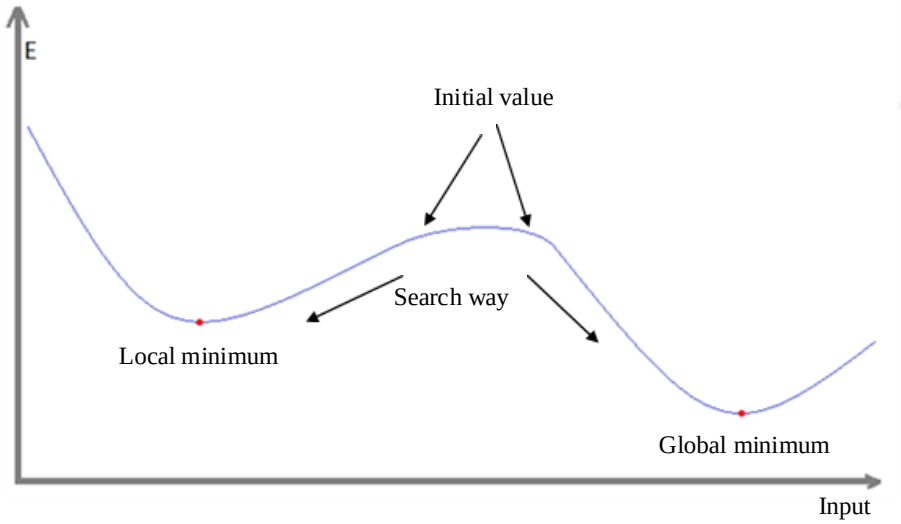

Figure 3.9. Local and global minimum for ANN's error function

Gradient descent is not the only method for optimizing the weights. There are different variations of the gradient descent algorithm. Adagrad, RMSProp, ADAM optimizer are a few of them (Murphy, 2016). Conjugate gradient, Quasi-Newton, Levenberg and Marquardt are other gradient based optimization algorithms (Nur et al., 2014). The method of updating the weights is called the learning rule. The four basic learning rules are error correction rule, Boltzmann learning rule, Hebbian learning rule and competitive learning rule (Basheer, I.A.,

200 C.E.). In recent years, metaheuristic algorithms have also been used to optimize weights in ANN (Nur et al., 2014).

3.2. CNN Architecture

The ANN has experienced two periods of stagnation since its discovery in the 1940s. The first of these is the period until 1960 when it could not solve complex classification problems (Keskenler & Keskenler, 2017). This problem has been solved with the discovery of multilayer perception networks. Another weakness, the processing cost, emerged from now on. The increase in the number of layers in the network has increased the number of parameters to be calculated. Another data determining the number of parameters to be calculated in the ANN is the size of the input vector. For example, the number of parameters to be calculated in a single layer of an ANN that processes MINST data, which is a 28x28 monochrome handwriting data set, is 784. If the image size is 64x64 and the number of color channel is 3, this number is 12288 (O'Shea & Nash, 2015). The solution to this problem has been found by increasing the processing capacity of computers since the 2000s and the use of graphic processors in the training of ANN (Murphy, 2016). The widespread use of the Internet, the processing of large data sets and high-quality images with ANN has turned it into a great challenge even for advanced computers.

CNN is a special type of ANN that promise solution to high training cost of ANN (Namatēvs, 2017). Although CNN generally has all the features of ANN, it promises to reduce the number of parameters with its additional layers. It also allows preserving the attributes of the data while reducing the size of the input data. Another feature of CNN is that it automates the feature extraction process that requires special experience (Figure 3.1). Although CNN is used in many different areas, the area associated with it is CV. For this, it would be appropriate to explain the CNN with the image classification problem.

While traditional image classification methods are based on descriptive analysis, deep learning-based classification methods are based on predictive analysis. Descriptive analysis aims to explain phenomena with mathematical models. Predictive analysis aims to reach a generalization by measuring independent events. The proposed model initially contains major errors. As the number of samples increases, the error value decreases. (Wang et al., 2018) (O' Mahony et al., n.d.).

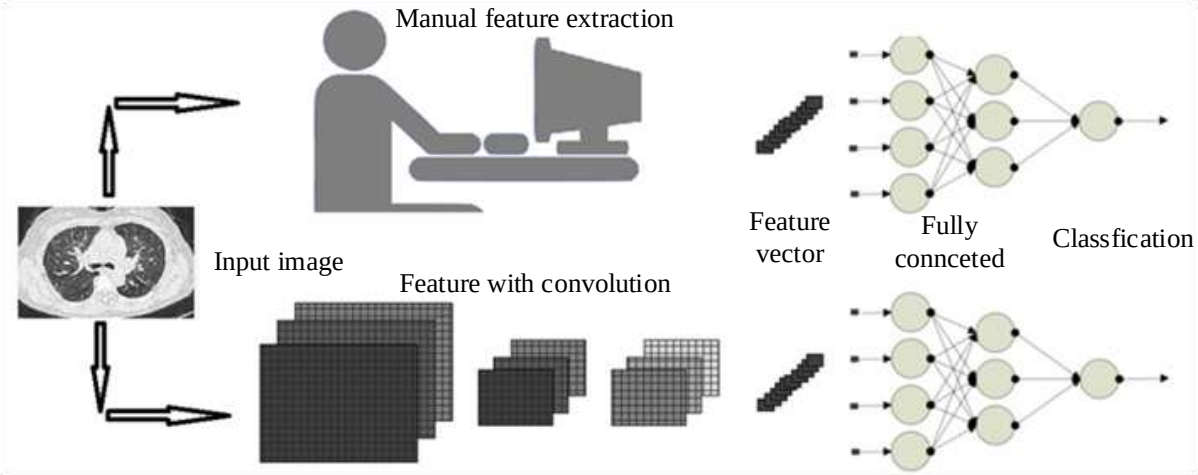

Figure 3.10. Traditional ANN and CNN computer vision flowchart

CNN consist of several ordered layers. Although different CNN architectures are used in different numbers of these layers, the simplest CNN structure in is shown (Figure 3.11). The layers described in this section are not considered independent layers for CNN. Really, Convolution + Pooling + Activation together are called only the Convolution layer.

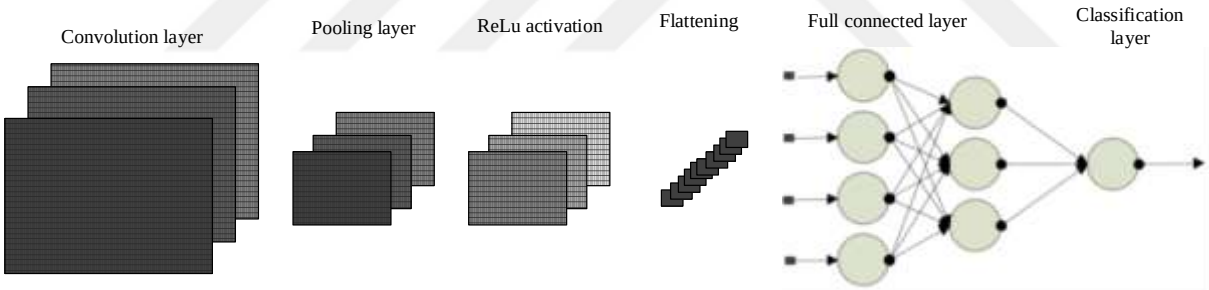

Figure 3.11. Building blocks of CNN

The other layer is the classification layer. It is also known as softmax layer. Here, these are explained under different headings for detailed explanation.

3.2.1. The Convolution Layer

Convolution is a special function that produces a new function when applied to two functions. The output of the convolution function is a measure of the overlap of functions given as parameters. In one dimension, the convolution between two functions is defined as follows:

$$g(x) = f(x) * h(x) = \int_{s=-\infty}^{\infty} f(s)h(x-s) \tag{13}$$

where f is main function and h is the function applied to f. The two-dimensional convolution between two functions is defined as follows (Namatēvs, 2017).

$$g(x,y) = f(x,y) * h(x,y) = \int_{s=-\infty}^{\infty} \int_{t=-\infty}^{\infty} f(s,t)h(x-s, y-t)ds\,dt \tag{14}$$

In binary computation systems, time series consists of discrete data. One-dimensional discrete convolution is defined as (Namatēvs, 2017):

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{15}$$

In machine learning applications, the input is usually a multidimensional array of data and the kernel is usually a multidimensional array of parameters that are adapted by the learning algorithm. If there is a two dimensional image I as an input, the two-dimensional kernel K_{mxn} has to be used. The convolution for two dimensions is as follows (Namatēvs, 2017).

$$S(i,j) = (I * K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n)K(i-m, j-n) \tag{16}$$

Since gray scale images are expressed in two-dimensional matrices, the convolution mentioned in the rest of the article is two-dimensional. The convolution for each element of the input matrix is equal to the sum of the products of overlapping elements.(Figure 3.12) Overlapped elements centre on the all elements of input matrix sequentially. For multi-channel input data, the convolution is applied independently for each channel (Wu, 2017). For coloured images, the convolution process is applied separately for each colour channel.

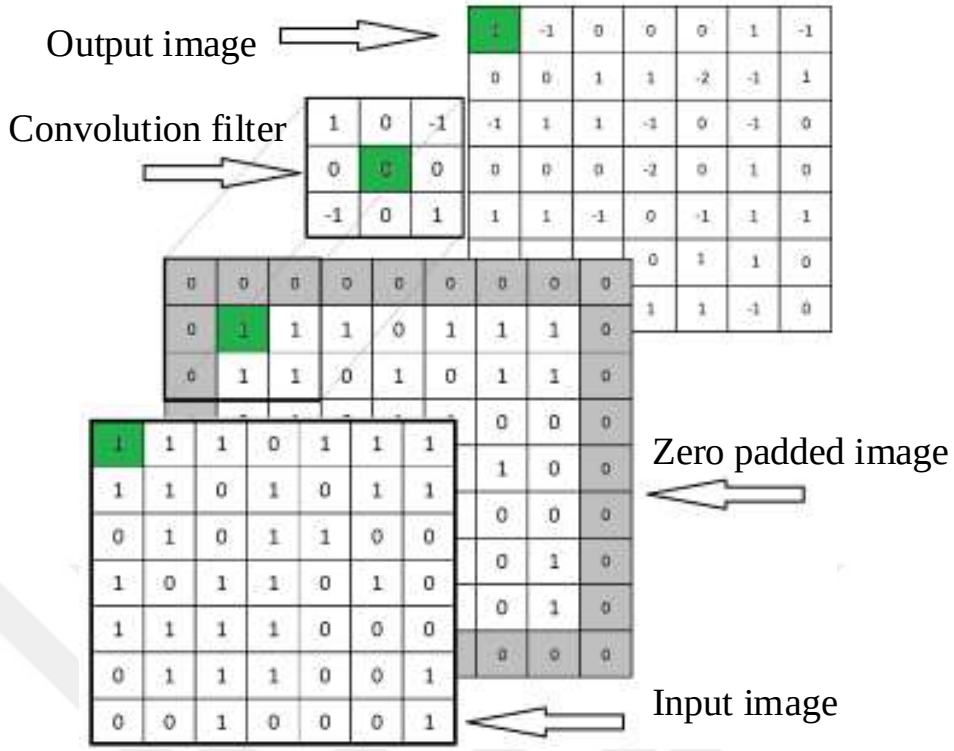

Figure 3.12. Dimensional fixing with zero padding in convolution

In manual feature extraction methods, selection of kernel matrix is essential. Effects of every kernel matrix on input matrix different. Some of these effects the edges and others affect the textures (Figure 3.13). Filters in the CNN structure contain random values at the beginning. During the training of the network, these filters are updated to reduce the learning error.

Filter matrices are mostly quadratic. The fact that the filter has a square and odd number size provides it to centering on a special element of the input matrix. Filter size is related to the size of the input image as well as the size of the feature of interest on the image (Murphy, 2016). Here, the original image is the input image in (Figure 3.13) The image has been resized in square form. The source of the original image is the COVID-CT-Dataset(Yang et al., 2020).

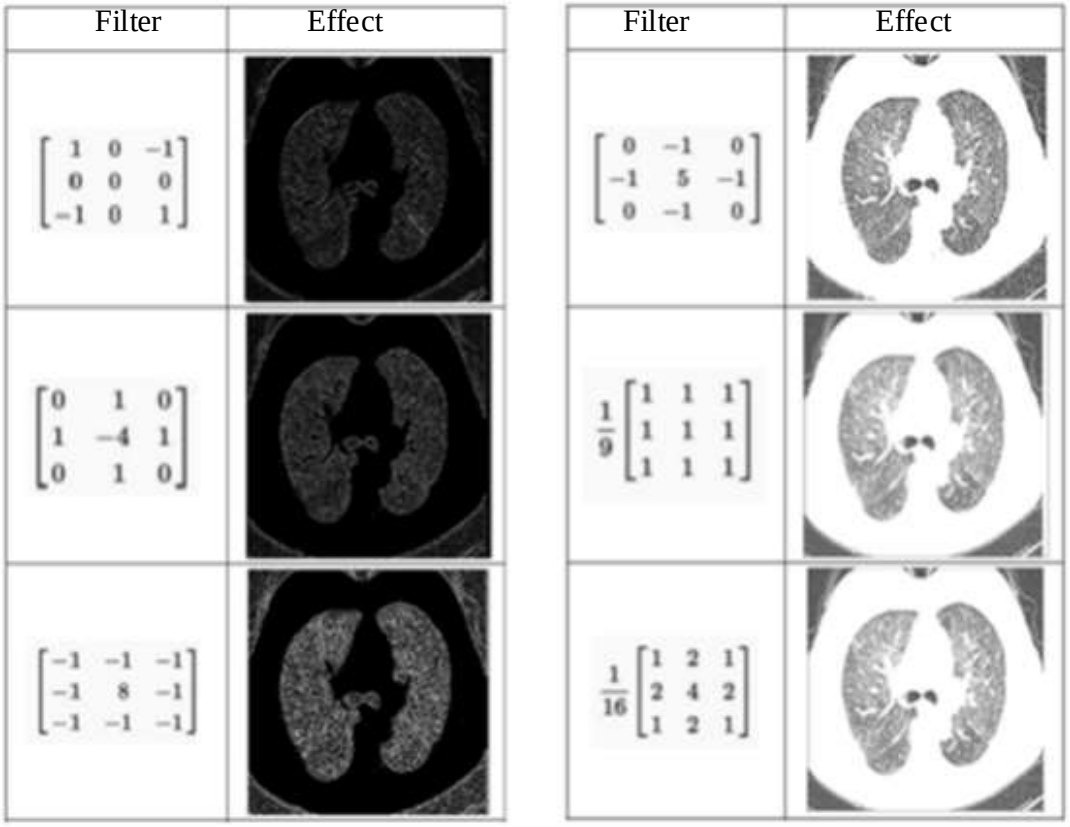

Figure 3.13. Effects of kernel matrix on image

One of the drawbacks of the convolution is the loss of information at the border of the image. For example, in (Figure 3.14), although the input size is 7, the output size is 5. Here input matrix and kernel same as in (Figure 3.12).

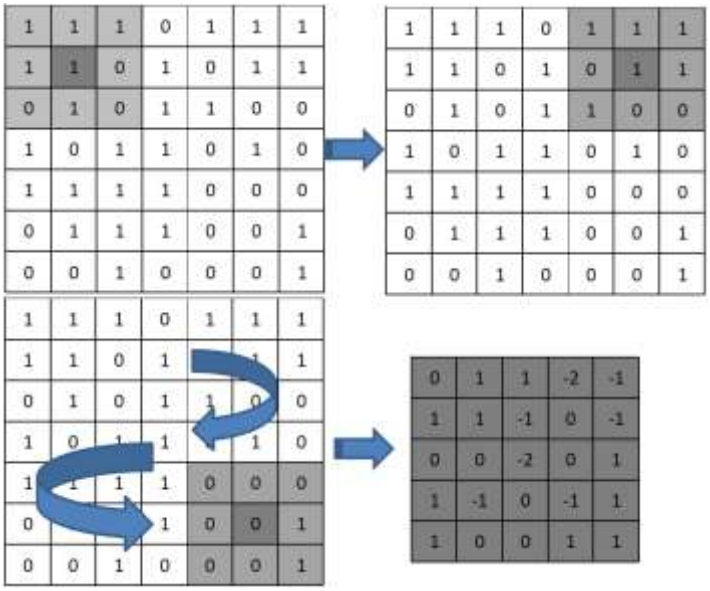

Figure 3.14. Data loss at border of image while convolution process

A very simple, yet efficient method to resolve the issue is to use zero-padding before convolution (Albawi et al., 2018). (Figure 3.12)

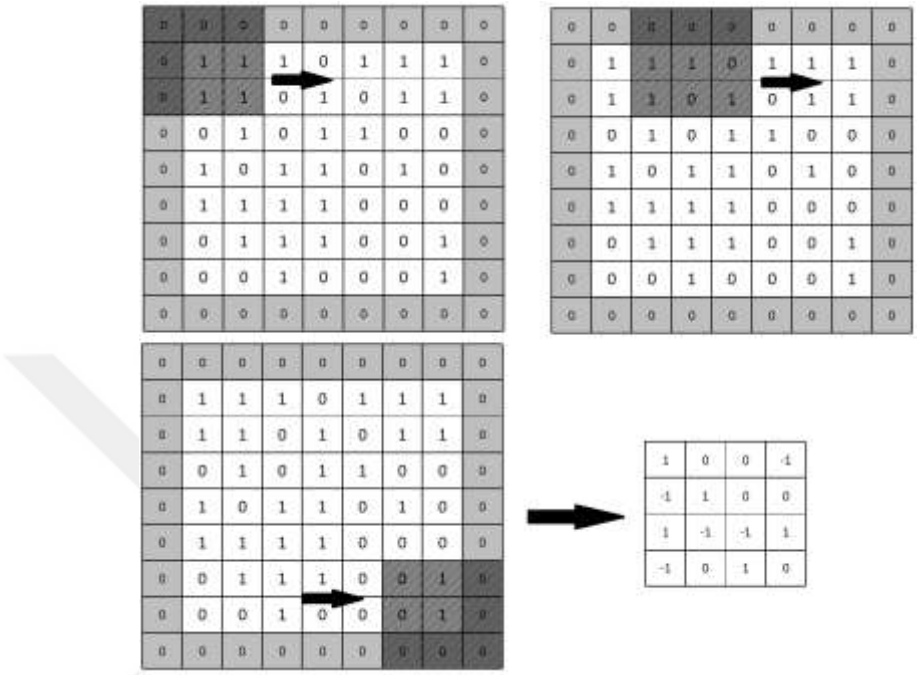

Figure 3.15. Convolution with stride=2

Stride is the distance between sequential pixels in convolution. If set stride is 1, then we would have a heavily overlapped receptive field producing extremely large activations. Alternatively, setting the stride to a greater number will reduce the amount of overlapping and produce an output of lower spatial dimensions (O'Shea & Nash, 2015) (Wu, 2017). Figure 3.15 shows the size reduction effect of stride. Here the matrix and filter are the same as in Figure 3.12. When convolution is applied, the size of the output image will be different from the input. This difference depends on the input image size, kernel size, amount of stride and zero padding (O'Shea & Nash, 2015) (Murphy, 2016). When zero padding is not applied, output size of convolution is as follows:

$$M = \frac{(V-R)}{S} + 1 \tag{17}$$

If zero padding is applied output size is as follows:

$$M = \frac{(V-R)+2Z}{S} + 1 \quad (18)$$

were V is the size of the input image. R is kernel size. Z is the amount of zero padding at the image boundary. S is the number of stride and M is the new dimension of output image. The amount of zero padding on each side, to keep the output size constant for a quadratic input matrix and the kernel is as follows(Wu, 2017),(Vedaldi et al., 2015):

$$Z = \frac{R-1}{2} \quad (19)$$

Usually a bias term is also added in the convolution layer. This ensures that the values are positive (Wu, 2017).

3.2.2. The Pooling Layer

Pooling is the selection of a pixel to represent a defined region on the image. This area, called the core, is usually quadratic. The movement of the kernel on the image is similar to the convolution operation. At the end of the pooling process, the average value of the pixels may change depending on the selection method, while the image sizes are reduced. In pooling, a common use is to select the pixel with the largest value. Here, the average pixel value increases. Distinctive features stand out. Max-pooling is one of the most common types of pooling methods. It partitions the image to sub-region rectangles, and it only returns the maximum value of the inside of that sub-region. One of the most common sizes used in max-pooling is 2×2 (O'Shea & Nash, 2015). Max pooling selects the most dominant feature for a particular region of the image. (Figure 3.16). Other types of pooling are mean pooling and minimum pooling(Namatēvs, 2017).

Both stride and pooling can be used for size reduction (Murphy, 2016). When pooling size and stride are equal each pixel is evaluated only once. Smaller stride value causes overlap. In this case, secondary features are suppressed. Some pixels can be ignored when the number of stride is greater than the pooling size.

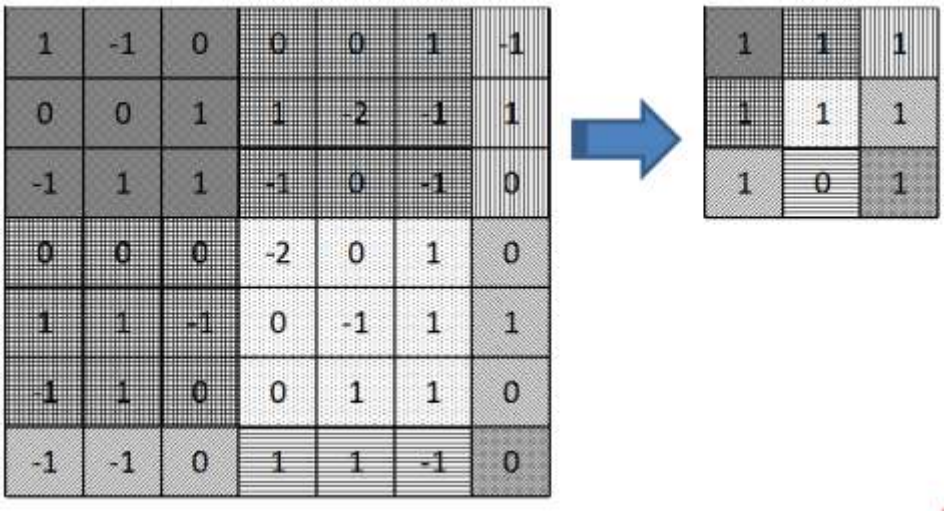

Figure 3.16. Max poling with size=3 and stride=3

The size change in the output image is similar to the equations in the convolution operation.

3.2.3. The Activation Layer

Linear regression model based on ANN establishes a linear relationship between input and output. There is no linear model for multilayer networks because of each layer generate a multiplier for the output. The output value increases in proportion to the number of layers. In reality, the value the network must produce is in a more limited range. This is sometimes a real probability value between 0 and 1. Sometimes it is a binary result. It is the activation function that provides this transformation(Table 3.1).

Table 3.1. Mostly used activation functions for CNN

Name	Sigmoid	Sofplus	Leak-ReLU	Tanh	Softsign
Expression	$\sigma(x) = \frac{1}{1+e^{-x}}$	$\ln(1+e^{x})$	$\begin{cases} a.x, & x < 0 \\ x, & x >= 0 \end{cases}$	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$\frac{x}{1+\lvert x\rvert}$
First derivative	$\sigma(x)(1-\sigma(x))$	$\frac{1}{1+e^{-x}}$	$\begin{cases} 1\ for\ \forall x \geq 0 \\ a\ othervise \end{cases}$	$1-tan^{2}x$	$1-x^{2}$

Saturated activation functions can not handle all real input fairly. They are very generous to them around zero. In far locations, the outputs are almost the same for each different input. This feature allows these functions to generate reliable probability values only for a limited range of inputs. Outside of this safe range is the saturation zone(Figure 3.17).

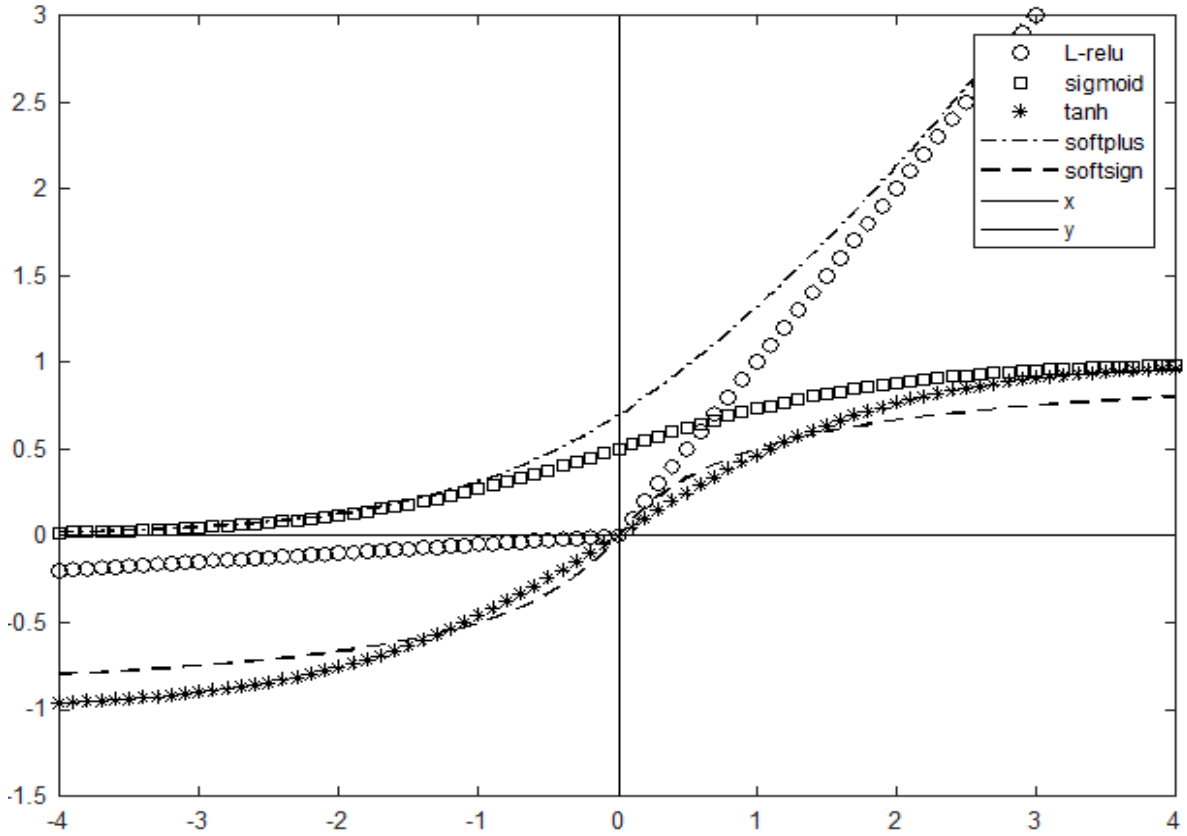

Figure 3.17. Activation functions for NN

Sigmoid, softsign and hyperbolic tangent are known for these properties. This feature causes vanishing gradient during back propagation. The ReLU function is resistant to this problem in the positive region.

The preferred activation functions in the middle layers and output layer of the CNN may differ. ReLU is usually used after the Convolution+Pooling layer(Namatēvs, 2017), (Yamashita et al., 2018).

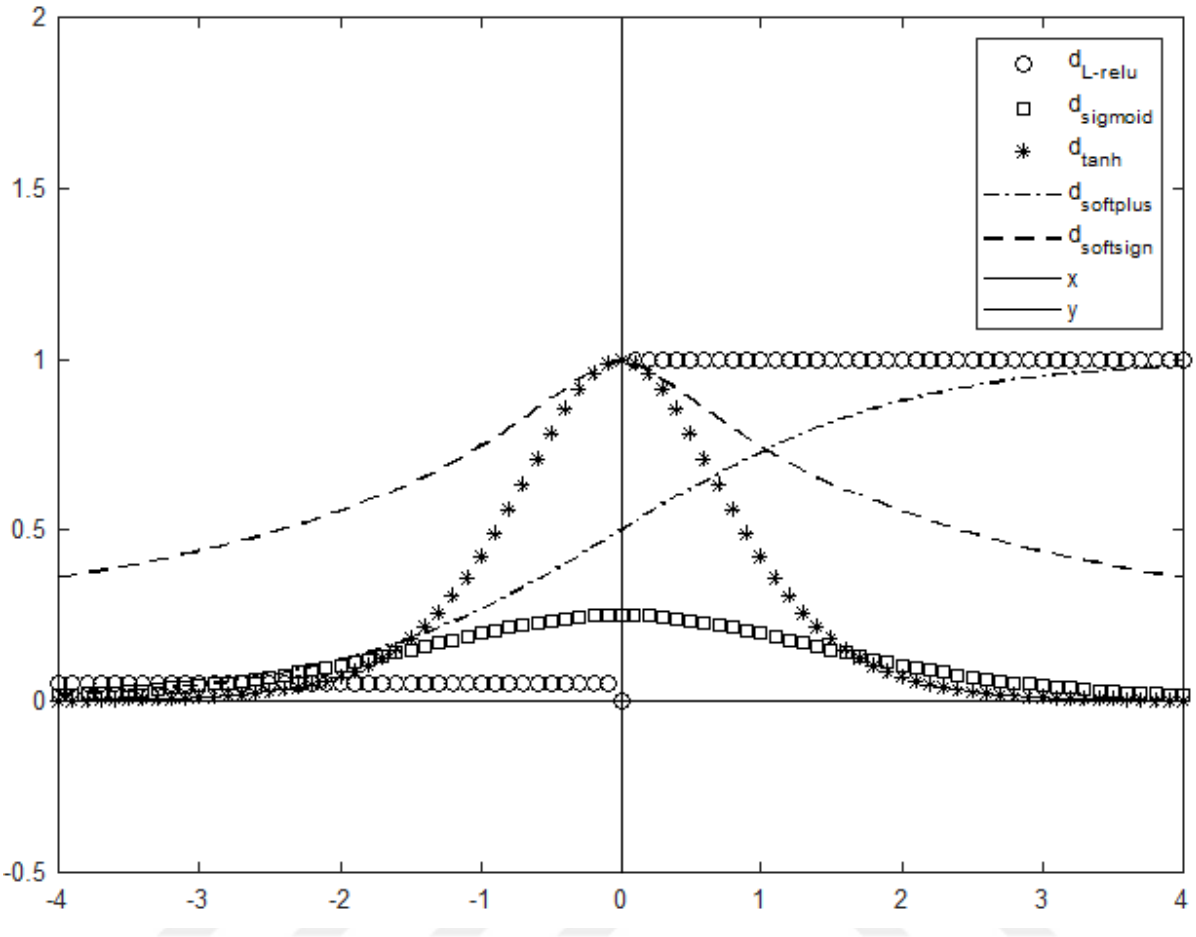

Figure 3.18. Derivatives of activation functions

In order to reduce the computational cost of the network, the activation functions and their derivatives should be easily calculated. The cost of activation functions is related to the mathematical operations involved in them and their derivatives. Activation functions containing logarithmic and exponential expressions are relatively more costly. Since the ReLU, softsign function and their derivative given in Table 3.1 do not contain exponential and trigonometric expressions, this function is a low-cost function. Activation functions must be continuous for all real number(Albawi et al., 2018) (Ding & Qian, 2018). If not used for solving linear problems, the output of activation functions must be nonlinear map to a certain limited range.

The fact that the triggering mechanism of human neurons shows similar characteristics to rectifier circuits known as electric rectifiers makes the ReLU function an activation function preferred in deep CV applications. ReLU activation function is applied after the pooling layer to prevent negative value entry to the next layers. This function keeps positive values while

converting negative values to zero. The properties of the ReLU activation function, referred to as a separate CNN layer in some sources, make it useful for CNN. The benefit of the ReLU function is that its value and its derivative can be calculated simply. It converges faster than other functions with saturation. Since it has a fixed derivative, it is easy to calculate and is not affected by the vanishing gradient problem (Lederer, 2021) (Murphy, 2016). On the other hand, this function also has disadvantages. Since its output is significantly positive, it produces a large bias value for the next layer (Ding & Qian, 2018). Since the left side derivative of the ReLU function is zero, it cannot update the weights for negative values. Different modifications of ReLU have been proposed as a solution to this problem. Some examples are LReLU, RReLU, PreReLU, ELU, MPELU (Ding & Qian, 2018) (Murphy, 2016) (Albawi et al., 2018).

3.2.4. The Flattening Layer

As mentioned in the ANN section, neuron outputs are produced by multiplying the input and weight values mutually and calculating the sum. Performing these operations in a sequential loop will increase processing time. In order to shorten the processing time, the dot product or matrix multiplication process are applied, which enables multiplication operations to be executed in parallel. Suitable input data for parallel calculation are in vector form and weights are in matrix form. In CV applications, input data are 2 or more dimensional tensors. Tensors are multidimensional data structures that can represent constant values(0D), vectors(1D), and matrices(2D). The first dimension of the tensor represents the rows(Height) in each color channel and the second dimension the columns(Width). The third dimension is depth. Color images represents as 3-dimensional tensors. $x \in R^{HxWxD}$ (Figure 3.19) Converting these data to vector form for adaptation to fully connected layer input is called flattening. In mathematic it also known as vectorization (Wu, 2017). Tensor in deep learning is more than three dimensions. Batch size adds a new dimension to tensor data. Presentation style for a batch data set consisting of N coloured images is $x \in R^{HxWxDxN}$. Similarly, convolution filters are also in the form of multidimensional tensors. Vectors are data structures in column form. It is expressed as a transpose of lines for ease of writing. The vector form of the tensor given in Figure 3.19 is given in Eq.20

$$V = [1\ 4\ 7\ \ldots\ 10\ 13\ 16\ \ldots\ \ldots 19\ 22\ 25\ \ldots\ ]^T \quad (20)$$

Here T represents transpose of row. As a result of the vectorization process, it is seen that the color values representing any pixel of the image are separated from each other. Still, the pattern relationship between these values is preserved.

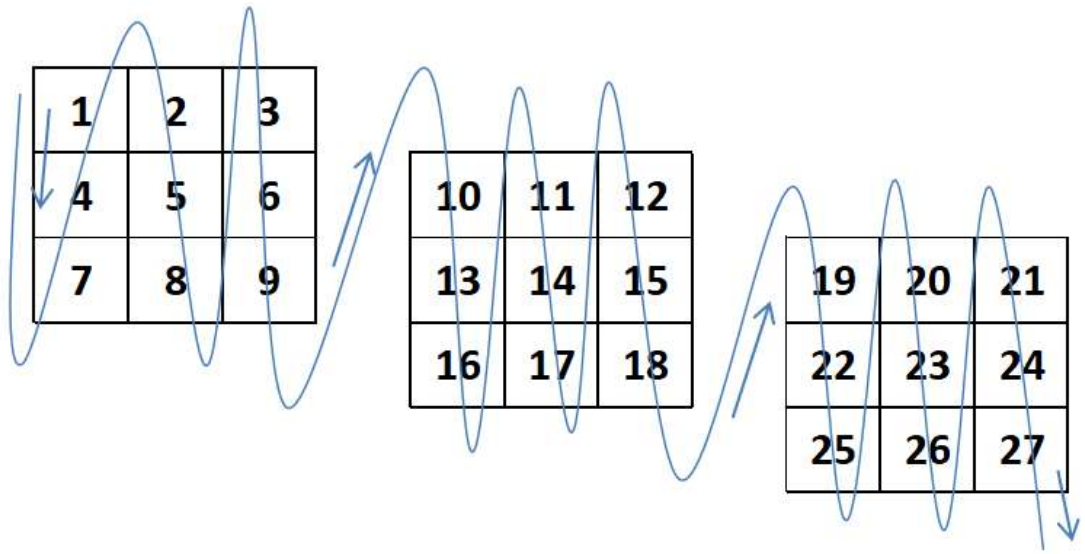

Figure 3.19. Tensor vectorization

The presentation of spatial properties as meaningful entries to the classification layer depends on the flattening layer. Various studies have been presented on this subject (Wang et al., 2018) (X. Ma et al., 2017) (Sadowski, 2017).

3.2.5. The Fully Connected Layer

The fully-connected layer is a similar to the way that neurons are arranged in a traditional NN. (Figure 3.3) Therefore, each node in a fully-connected layer is directly connected to every node in both the previous and in the next layer (Albawi et al., 2018). The number of fully connected layers increases the number of parameters to be calculated as well as increasing the depth of learning.

Determining the number of hidden layers is a critical task in CNN architecture. Low layer number does not allow learning complex patterns. A high number of layers can lead to memorization rather than learning. This also increases the training time cost. The number of hidden layers is decisive for the performance of the network. There is no magic formula for determining the optimal number of hidden layers. When you add 1 layer more than the number you think is the ideal number, it is difficult to predict how the success of the

classification and the time cost will change. It can be heuristically said that the number of layers is related to the number of input and output layers and the number of training parameters. Hidden layer size formulas based on these parameters have entered the literature (Basheer, I.A., 200 C.E.).

3.2.6. Output Layer

CNN's output layer is similar to ANN as its fully connected layer. Although the output layer is fully connected to the fully connected layer, the number of neurons and the activation function may differ. The number of neurons in this layer and the activation function depend on the purpose of the network. The sigmoid function is useful in binary classification networks, and the softmax function is useful in multi-classification networks(Yamashita et al., 2018).

3.2.7. Number of Parameters and Computational Complexity for CNN

An ANN should be trained in a reasonable time with limited hardware as well as its success. One of the hardware constraints is processor speed. One of the hardware constraints is processor speed. The number of calculation in each iteration depends on the number of learnable parameters determined as follows.

$$P = (C.R.R + 1).F \tag{21}$$

where C is number of channels in input dat, in this example it is equal to 1. R indicates the dimensions of the squared filter and F is the number of filters. In addition, a bias value is added for each filter.(Murphy, 2016) The stride S and padding Z values affect the convolution output sizes. These parameters have no effect on the number of trainable parameters. Activation function, which is not included in this model, and pooling layer also does not contain learnable parameters.

A simple CNN model is presented in Table 3.2 with the number of learnable parameters calculated according to Eq:21.

Table 3.2 A basic CNN model for learnable parameter counting

LAYER	INPUT SIZE (V)	FILTER SIZE(R)	STRIDE (S)	PADDING (Z)	FILTER COUNT(F)	OUTPUT SIZE(M)	PARAMS. COUNT(P)
CONVOLUTION	32x32x1	9x9	1	0	4	24x24x4	328
POOLING	24x24x4	2x2	2	0	0	12x12x4	0
CONVOLUTION	12x12x4	7x7	1	0	3	6x6x4x3	591
POLING	6x6x12	2x2	2	0	0	3x3x12	0
CONVOLUTION	3x3x12	3x3	1	0	3	1x1x12x3	111
FULLY CONN.	1x1x36	1x1	1	0	3	1x1x3	108
CLASSIFICATION	1x1x3	1x1	1	0	2	1x1x2	6

The first convolution+pooling layer of the CNN in Table 3.2 is seen in Figure 3.20

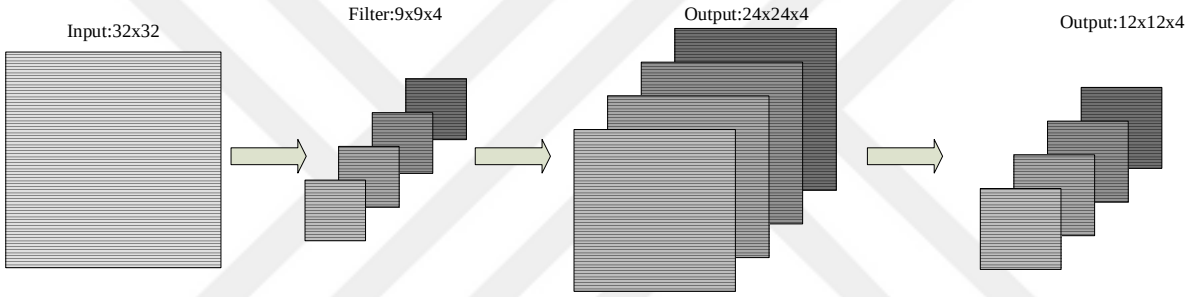

Figure 3.20 First convolution + pooling layer of the model

Second convolution+pooling layer contains 591 trainable parameters.(Figure 3.21)

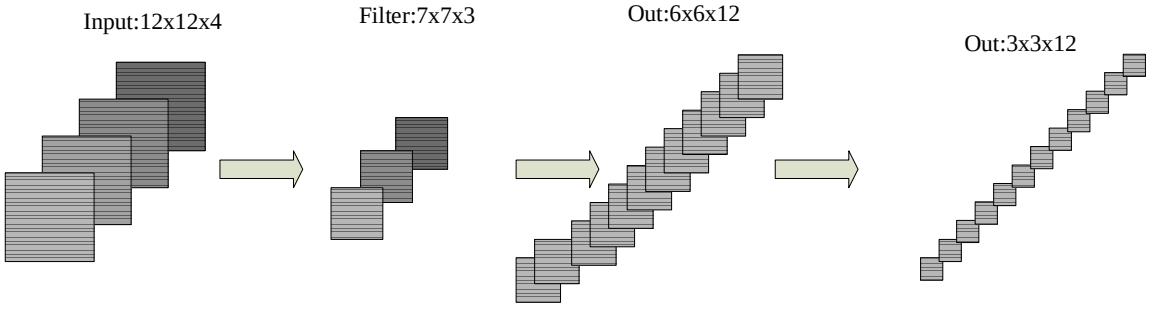

Figure 3.21 Second convolution + pooling layer of the model

Third convolution layer is also known as the flattening layer, as the last convolution layer makes the input sizes equal to 1(Figure 3.22).

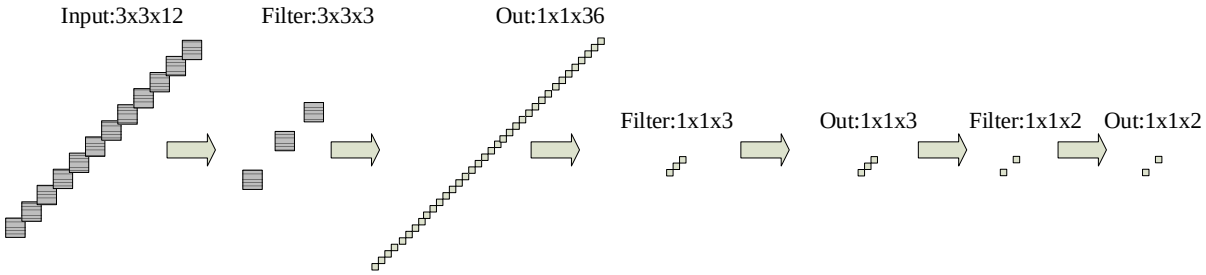

Figure 3.22 Last three layer of the model

Another constraint for ANN is hardware memory. The memory requirement for CNN is not only due to learnable parameters. Input data and error values for the batch should also be stored. Also, gradients need to be stored for back propagation. The number type of hardware also determines the memory requirement. Considering all these variables, the concept of memory footprint is more appropriate for a network rather than the exact amount of memory. The memory footprint gives an idea of the memory requirement of the network.

3.3. Fuzzy Logic

Fuzzy logic theory is an alternative to classical logic theory(Işıklı, 2007). Classical logic includes operations defined on classical sets containing discrete and definite membership functions. Fuzzy logic, on the other hand, includes operations defined on continuous and uncertain sets.

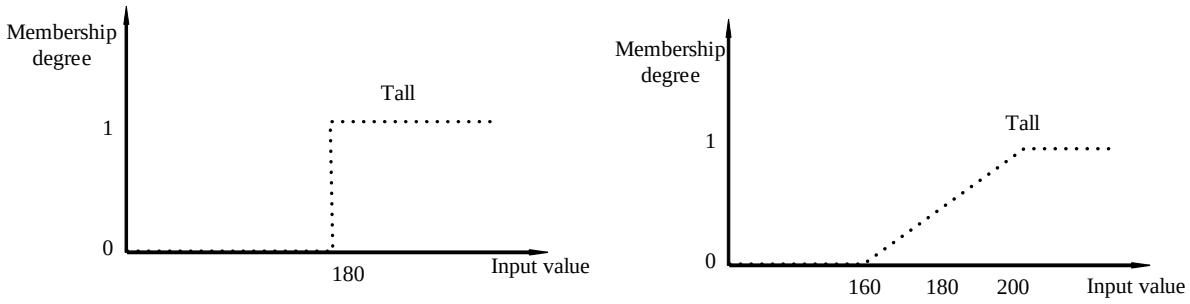

Figure 3.23. Classical and fuzzy set membership functions.

While membership is expressed as “yes” and “no” on classical sets, membership in fuzzy sets is expressed with real numbers between 0-1. In this respect, fuzzy logic provides control mechanisms suitable for processing analog signals produced in the physical world. In addition, fuzzy logic provides mechanisms to represent verbal attributes(Sivanandam et al., 2007). Verbal attributes, like fuzzy sets, contain uncertainty. For example, there is no definite

limit for the minimum and maximum values of the concept of "Tall"(Figure 3.23). Arithmetic-logical operations on verbal expressions provide suitable development environments for humanoid systems.

Fuzzy systems are created by operations defined on fuzzy sets. In fuzzy sets, the state of belonging to the set for any input is expressed with continuous membership functions. The membership functions can be triangle, trapezoid, bell, gaus, s-shape and z-shape. Fuzzy set operations are defined similarly to classical set operations(Sivanandam et al., 2007). The union operation returns the larger of the two membership functions(Figure 3.24)(Eq:22).

$$F(x) = \max\left(F_A(x), F_B(x)\right) \tag{22}$$

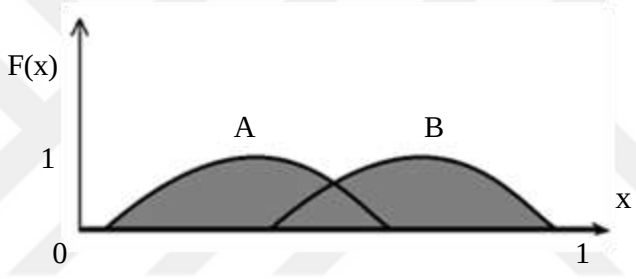

Figure 3.24. Union operations in fuzzy sets

The intersection operation returns the smaller of the two membership functions(Figure 3.25) (Eq:23).

$$F_{A\cap B}(x) = \min\left(F_A(x), F_B(x)\right) \tag{23}$$

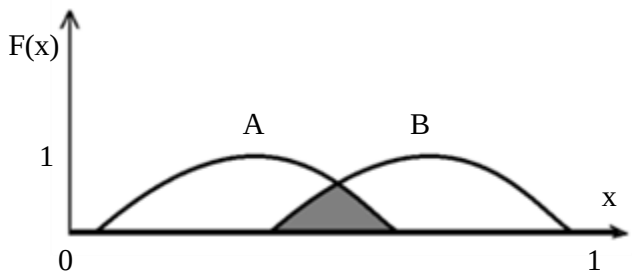

Figure 3.25. Intersection operations in fuzzy sets

The complement operation is the value that completes the function value to 1(Figure 3.26)(Eq:24).

$$F'_{A\cap B}(x) = 1 - F_A(x) \tag{24}$$

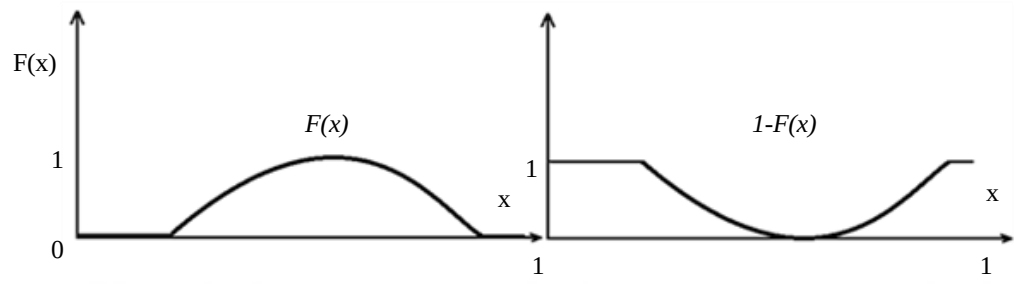

Figure 3.26. Complement operations in fuzzy sets

A fuzzy systems consist of fuzzy input membership functions, fuzzy rules and fuzzy output().

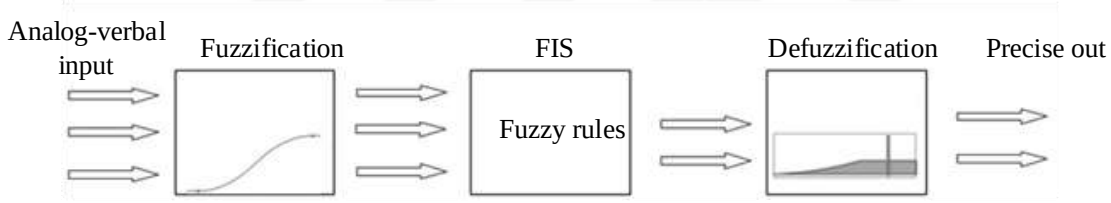

Figure 3.27. Basic buildig block of fuzzy sysetms

The fuzzy output generates a mechanism based on fuzzy inputs is called as Fuzzy Inference System (FIS). FIS, which is the basic mechanism of the fuzzy logic system, is a rule-based decision making process. Rules work with the IF....THEN..... mechanism. In systems with more than one rule, the interaction of the rules is done with AND, OR connectors(Sivanandam et al., 2007).

The output of the FIS usually creates a segmented and continuous graph, depending on the input-output and rule set. An exact result for a particular entry is based on the selection of a suitable point of this graph.

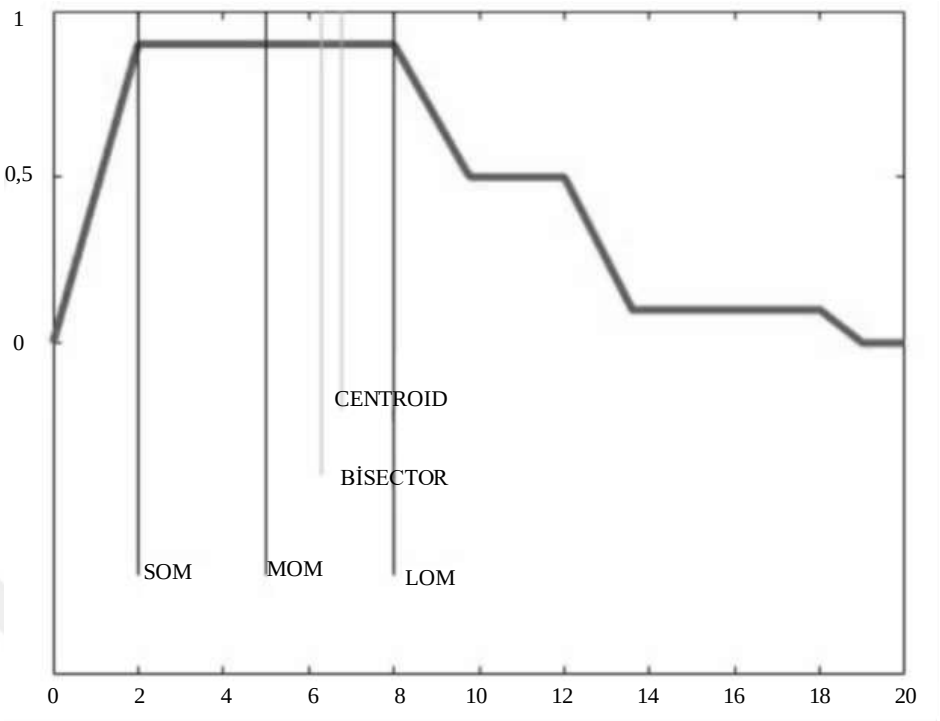

Figure 3.28. Fuzy Toolbox defuzzification methods[1]

Returning a value to represent this output is called as defuzzification. The defuzzification functions provided by Matlab Toolbox are given in (Figure 3.28).

3.4. Preparing Dataset

Data set selection is vital for all ANN types that derive their capabilities from their design and the data set they were trained with. This competence is required for both quantity and quality. The quantity is that the training dataset contains enough samples to ensure sufficient success for all test possibilities. Quality is that the data set can represent all possible test data. In this thesis, since the data consists of images, the preparation of the data is in the following three sections.

3.4.1. Imaging Thecniques for Diagnosos of Pandemic

During the Covid-19 epidemic, it has been seen that the best effective way for prevent the pandemic is preventing the spread of the disease among humans. It is important to quickly diagnose infected people to prevent contamination. Symptoms caused by the disease in the

[1] https://uk.mathworks.com/help/fuzzy/defuzzification-methods.html

lung tissue can be diagnosed by specialist radiologists on biomedical chest images. Studies have shown that deep learning methods can accelerate the diagnosis by helping radiologists in the diagnosis of the Covid-19. We can divide the Covid-19 diagnostic studies into three groups with deep learning methods from biomedical chest images. These are diagnosis from X-ray images (Narin et al., 2020) (Ismael & Şengür, 2020) (Apostolopoulos & Mpesiana, 2020) (Dansana et al., 2020) (Bressem et al., 2020) (Marques et al., 2020) the CT images (Yasar & Ceylan, 2021) (Ates et al., 2020) (Zhang et al., 2020) (Tilborghs et al., 2020) (Yan et al., 2020) (Yan et al., 2020) (J. Ma et al., 2021) (Pham, 2020) (Singh et al., 2020) and MRI images (Ates et al., 2020). Of these, the CT and X-Ray images are commonly used imaging techniques in the diagnosis of pandemics (Bressem et al., 2020).

3.4.1.1. X-ray Imaging

X-Rays were discovered by Wilhelm Konrad Röntgen in 1895 during his studies with Hittorf-Crookes tubes. He named these rays X-rays because of their unknown properties.

Figure 3.29. Chest X-ray image(Ilangovan et al., 2016)

This technique is based on the principle of reflection of X-rays emitted from a source after passing through body tissues, on a film or screenFigure 3.30. Since X-ray imaging is based on the one-way shading technique, the images are two-dimensional. Dense textures in the body hide each other (Ilangovan et al., 2016). By using beams of different intensities in imaging, X-rays can be used in many areas other than medicine, such as security and quality control. (Sun et al., 2012).

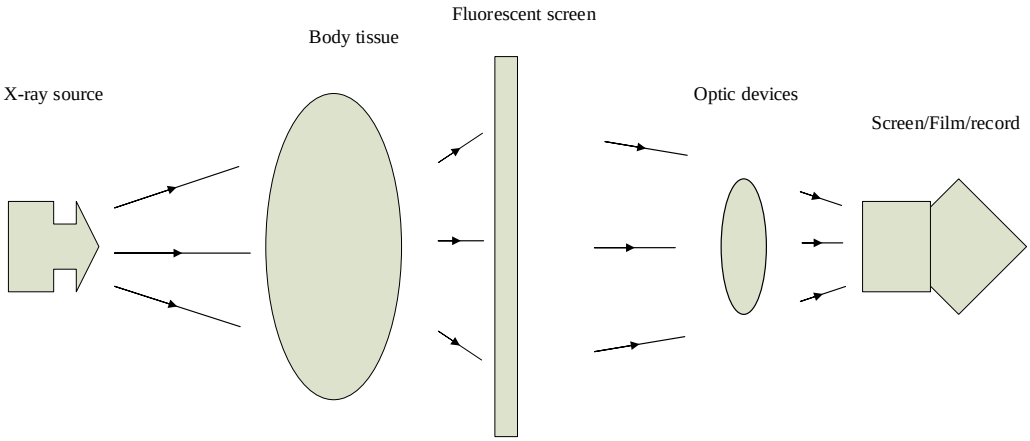

Figure 3.30. Overview of the basic X-ray imaging system

This X-ray, which has been used as a basic imaging in the medical field for many years since its discovery, is the basic phenomenon on which the CT imaging is based.

3.4.1.2. Computed Tomography Imaging

Tomography is derived from two Latin words "tomus" meaning slice and "graphein" meaning recording. Hounsfield and Cormack, who developed the CT idea unaware of each other, won the Nobel Prize in psychology and medicine in 1979 (Sun et al., 2012). Godfrey Hounsfield build first prototype the CT scanner. In the first image produced with the CT, the human brain was imaged. (Ilangovan et al., 2016).

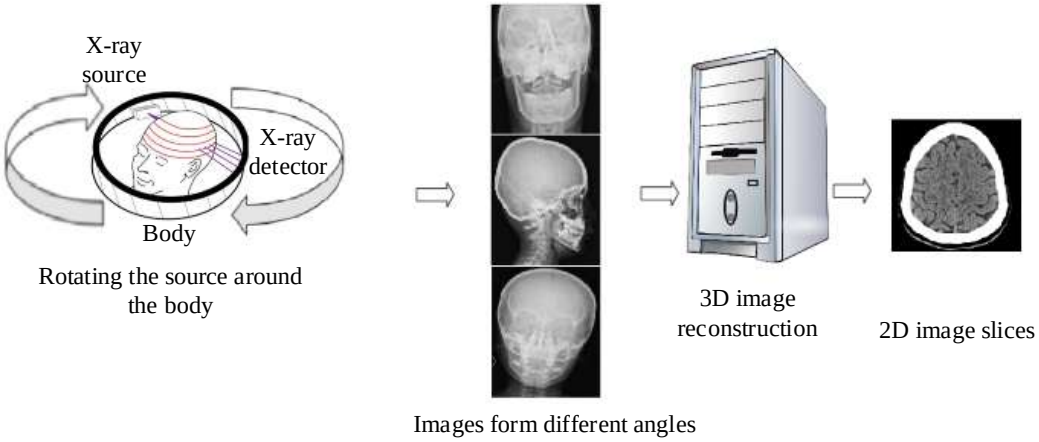

Figure 3.31. CT image reconstuction[2]

[2] The figure is a composition of images compiled from different sources

The CT is a three-dimensional imaging technique based on the principle of absorption of x-rays by matter. Rays emanating from an x-ray source (cathode tube) are absorbed depending on the density of the material in the medium through which they pass. The x-ray beam leaving the matter carries the density information of the medium. This information is detected by a detector and stored as digital data. Each point of three-dimensional matrix is called a voxel (**Hata! Başvuru kaynağı bulunamadı.**). The intensity value of a voxel is proportional to the mount of radiation absorbed at that point. The beam passing through this point also passes through many voxels before and after it.(**Hata! Başvuru kaynağı bulunamadı.**)

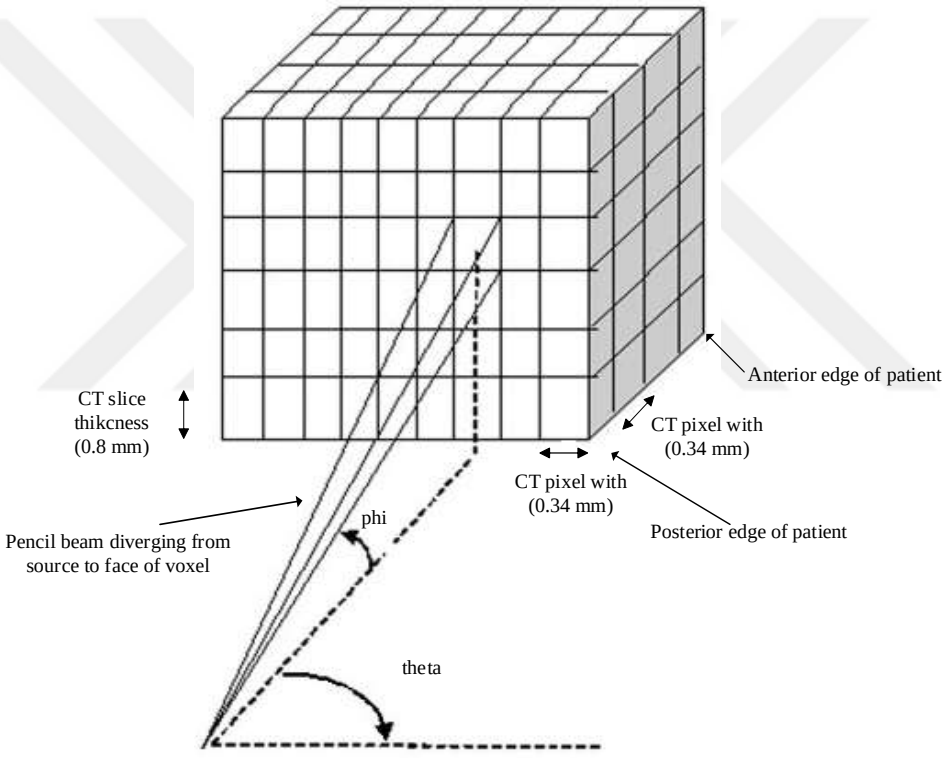

Figure 3.32. The value of voxels depends on the amount of x-rays they absorb[3]

To differentiate the intensity value of a voxel from others, the information of beams passing through that voxel at different angles is used. The voxel value of the three-dimensional images is obtained by calculating the stored angle and density data by the computer with

[3] (Moore et al., 2011).

different transformation algorithms such as Radon, Sidon etc.(Sun et al., 2012) (Moore et al., 2011).

Figure 3.33. A slice of chest CT image[4]

This is called image reconstruction. Scanning is done around an axis and at fixed intervals. Voxel information is visualized as two-dimensional slices perpendicular to this axis (Ilangovan et al., 2016). (**Hata! Başvuru kaynağı bulunamadı.**)

3.4.1.3. Magnetic Resonance Imaging

NMR was discovered by Felix Bloch and Edward Purcell. This was the first step in the discovery of MRI. (Ilangovan et al., 2016).

Subatomic particles such as protons, neutrons and electrons have a phenomenon called spin. Particles with a spin number greater than zero have a measurable magnetic moment associated with it. MRI is an imaging technique based on detecting spins of these particles. The proton is the subatomic particle with the largest magnetic moment, and MRI imaging is basically based on proton MRI. The proton in a magnetic field moves around the magnetic field depending on the magnitude of the magnetic flux and the gyromagnetic ratio (Figure 3.34).

[4] https://upload.wikimedia.org/wikipedia/commons/1/12/High-resolution_computed_tomograph_of_a_normal_thorax%2C_axial_plane_%2872%29.jpg

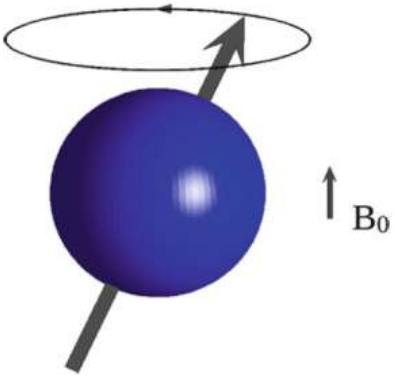

Figure 3.34. Magnetic moment of proton in a magnetic field[5]

The electromagnetic radiation generated during this movement is recorded by the detectors. These data are used in reconstruction of MRI images by different reconstruction techniques (Doneva, 2020). The density of the hydrogen atom with a proton in the human body makes MRI especially important for imaging soft tissues.

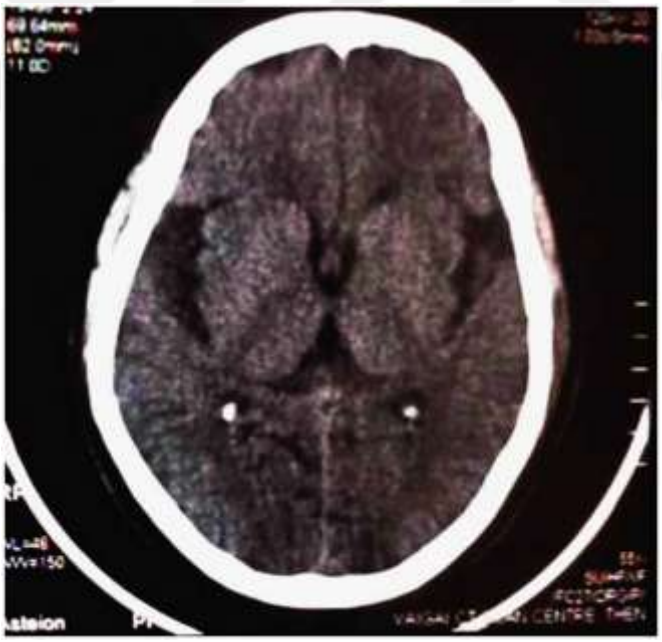

Figure 3.35. MRI image of a brain[6]

The advantages of MRI over the CT are that the image contrast can be adjusted and the image section can be taken in different planes.(Ilangovan et al., 2016).

[5] Möllenhoff et al., 2012
[6] Ilangovan et al., 2016

3.4.2. Datasets for the Pandemic Dieeases

The success of deep learning algorithms depends on training them with well-chosen data. In this section, publicly available chest image datasets are introduced.

3.4.2.1. Chexpert Dataset

The CheXpert dataset consists of 224316 chest X-ray images obtained from 65240 patients. Data is assigned to one of three different classes for 14 features. These are positive, negative and uncertainty states. In this respect, it is a data set suitable for multiple classifications. The confirmation set of 200 data was manually labeled by 3 board-certified radiologists. The data is labeled with the created labeling tool. The performance of 3 different radiologists in detecting 5 pathologies in 500 samples selected from the data set was compared. Since the dataset was obtained at Stanford Hospital between 2002 and 2012, it does not include data on the Covid-19. However, due to the different lesions it contains, it is suitable for multiple classification studies. Since the dataset contains lesions similar to the Covid-19 symptoms, it is suitable for use in increasing the success of the learning algorithm(Irvin et al., 2019).

3.4.2.2. SARS-CoV CT-scan Dataset

This dataset consists of 2482 the CT images obtained from the records of the Brazilian Sao Paulo hospital. 1252 of these images belong to 60 patients diagnosed with Covid positive. The remaining 1230 images are Covid negative but belong to 60 patients with different lung diseases. An explainable deep learning method (xDNN) was used to test the dataset. The data set prepared in this thesis has been shown to be very important in the diagnosis of the Covid-19 from the CT images. In addition, the recommended xDNN model has been shown to be highly successful in diagnosing the Covid-19(Soares et al., 2020).

3.4.2.3. COVID-CT-Dataset

In this thesis, an open-source dataset consisting of 349 Covid positive and 397 Covid negative data belonging to 216 patients is presented. This dataset has been shown to be useful for developing AI-based diagnostic models. The data set includes gray level and rgb images. It should also be taken into account that the images have different file formats (Yang et al., 2020).

3.4.2.4. Covid-19 Image Data Collection

This dataset consists of front view chest X-ray images. Images were manually obtained from various publications and web-based repositories. It contains 679 images of 412 people from 26 different countries. The data is presented with code that allows automatic loading by machine learning algorithms. A metafile containing additional patient information is also presented. In addition to the Covid finding, this file contains various data such as RT-PCR test result, survival time, intubation status, etc. In this respect, it is a suitable data set to produce predictions about the need for an intensive care unit, the patient's future condition, survival rate, etc(Cohen et al., 2020).

3.4.3. Data Prepocessing

Images provided from different sources may have different standards. The source of this difference is the environment from which the image is obtained, the features of the image acquisition tool, image recording standards etc. may be. Since deep learning algorithms imitate the human brain, the success of classifying quality images will be higher. Because the distinctive features of quality images are more clear. For some images, the edges are distinctive, for others, the textures are more distinctive. Training and test images must have common standards for CNN's success. Some of the pre-processing techniques that can be applied to meet this standard are as follows (Basheer, I.A., 200 C.E.). It is important that the preprocessing applied to the training data is applied to the validation and test data as well.

3.4.3.1. Resizing

All input images must be the same size. Since different sizes will produce different output, it causes size mismatch in vector operations. The number of channels of the images should also be equal. For example, two images that both appear gray might have one RGB and the other gray level encoding. In this case, using RGB format will increase the training cost of the network. This causes also incompatibility with the convolution filter numbers. On the other hand, the image dimensions must be appropriate for the network to learn in a reasonable time. For example MNIST dataset which mostly used in image processing algorithm is normalized to a size of 28x28 pixels.[7]

[7] http://yann.lecun.com/exdb/mnist/

3.4.3.2. Image Data Formats

Different image standards may use different number formats. Like integers, real numbers, positive and negative values. All images must be in the same number format and bit width. Using different formats together trivializes the attributes of some images. For example, when an 8-bit unsigned number (0-255) and a real number (0-1) are processed together, the real number becomes meaningless. Many data normalization methods have been proposed to overcome this problem (Basheer, I.A., 200 C.E.).

Figure 3.36. Color level match between training and test images

The color values of the meaningful textures of the picture should be coded similarly in the training and test data. For example, if a feature is white-coded in the training data for a gray-level image, similar coding should be used for the test data (**Hata! Başvuru kaynağı ulunamadı.**). If necessary, the complement of the picture can be taken.

3.4.3.3. Histogram adjusting

The uniform distribution of image value is important for image features. Unbalanced distribution suppresses properties.

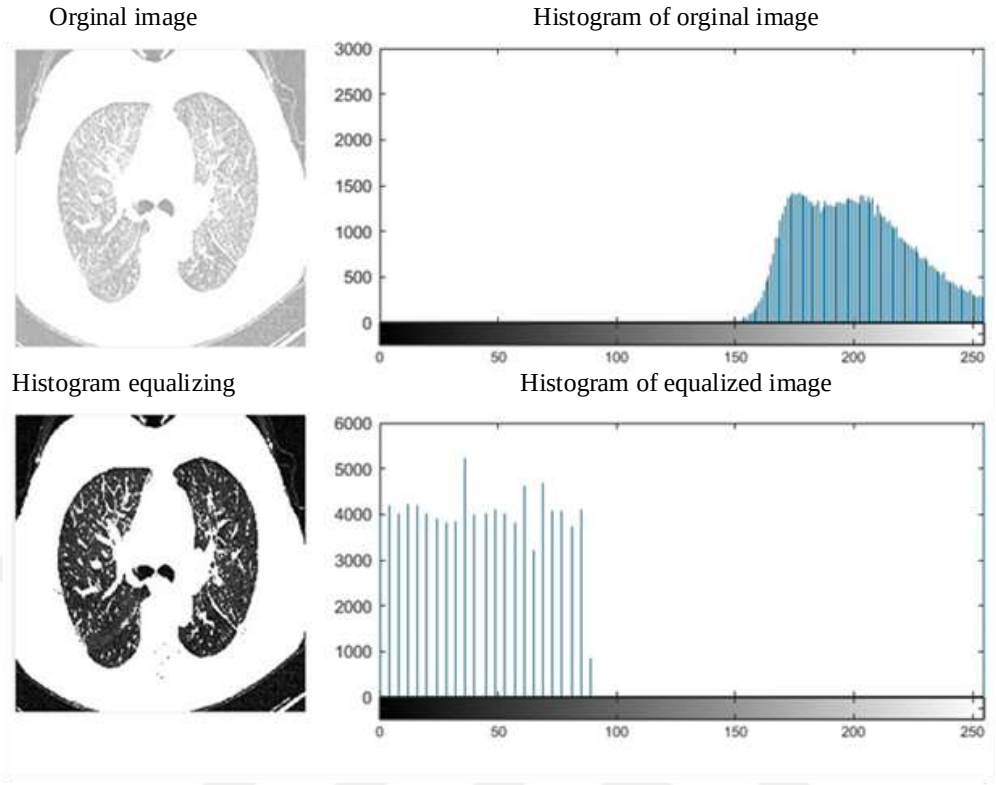

Figure 3.37. Effects of histogram equalization

Histogram equalization aims at uniform distribution of the histogram of the image. If the histogram value is desired to be in a certain range, histogram stretching can be applied.

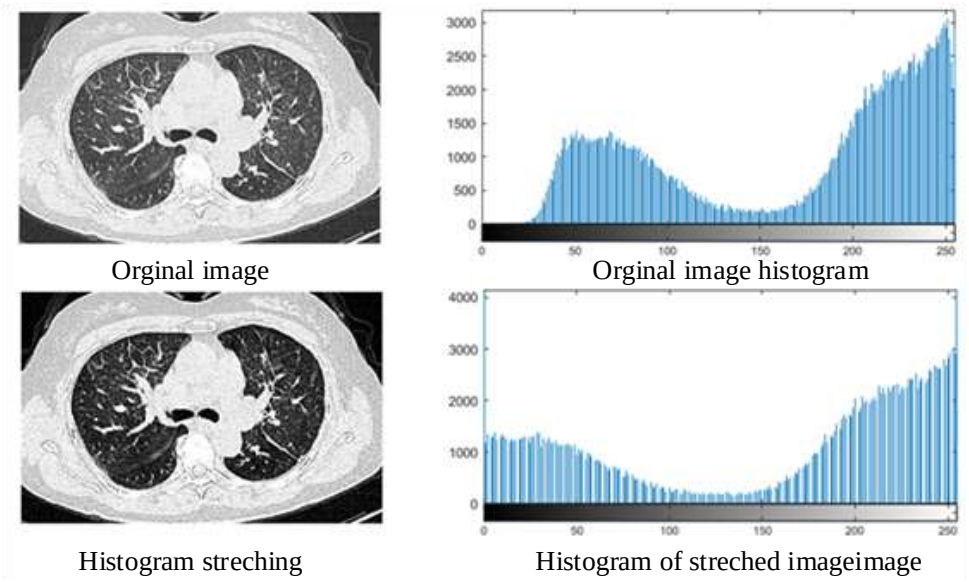

Figure 3.38. Effects of histogram stretcing

Histogram stretching may not be suitable for automatic image preprocessing, as the histogram distribution of images from different sources will be different.

3.4.3.4. Mean Normalization

A suitable value can be added to all pixel values to adjust the average level of the image. We can also do this to get rid of zero values (Sudeep & Pal, 2016).

Figure 3.39. Effects of mean normalization

Hata! Başvuru kaynağı bulunamadı. shows the effect of mean normalization on the image.

3.4.3.5. Gamma Correction

Image clarity is as important for CV as it is for human vision. Image brightness is one of the requirements for clarity. Increasing the brightness linearly can cause blurring of features in some areas of the image. With an exponential expression, the image brightness distribution can be separated nonlinearly. Thus, dark and bright areas in the image are illuminated at different rates. Such an effect can be achieved with (Eq:25).

$$x_{out} = x_{in}^{\gamma} \tag{25}$$

The gamma value is between 0 and infinity. If the gamma is larger than 1, the image brightness increases, otherwise it decreases(Singnoo & Finlayson, 2010).

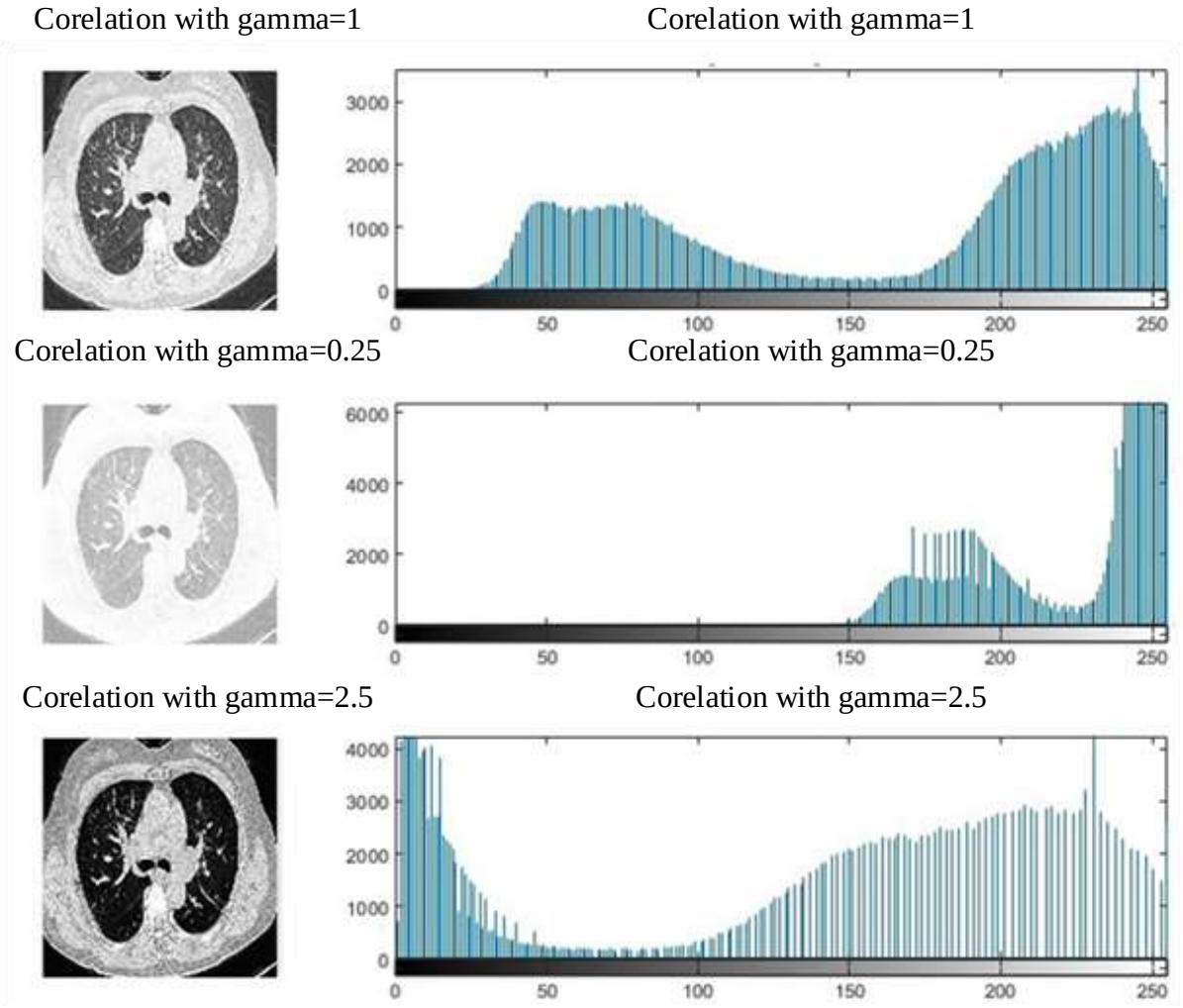

Figure 3.40. Correlation with different gamma values

Gama value is equal to 1, correlation is linear. Figure 3.40 shows the effect of gamma correlation on the image.

3.4.4. Data Augmentation

Being trained with insufficient data causes overfitting in deep learning algorithms. Creating large sets of biomedical images has several constraints (Zhang et al., 2020) (Murphy, 2016). These are the rarity of diseases, patient privacy, the need for expertise to label images, and manual processes in the production of images. Various data augmentation methods are used in deep learning (Yasar & Ceylan, 2021). Data enhancement methods can be listed under the following headings. Geometric transformations, color space transformations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, GAN-based

augmentation, neural style transfer, and meta-learning schemes (Shorten & Khoshgoftaar, 2019). Every method cannot be useful for every image set.

The image preprocessing methods mentioned in Section 3.3.3 are also usable for data augmentation. For example, the new image generated from an image using the histogram equalization is a new data highlighting textural features. Images produced with different gamma correlation values or scaling images can also be used for similar purposes (Zhang et al., 2020).

In the other hand conventional feature extraction/emphasize methods also suitable for data. Several of them are Local Binary Pattern, Local Entropy, Gray-Level Co-Occurrence Matrix (Yasar & Ceylan, 2021). Since CNN is a classifier based on spatial features, the use of data reproduced in different transformation spaces should be carefully planned. How meaningful would it be to apply an image and its Fourier transform as input to CNN in the same training set?

3.4.4.1. Geometric Transformations

Flipping**:** Mirrors the image along the vertical and horizontal axis. Horizontal flipping is more commonly used. This method has proven useful in datasets such as CIFAR-10 and ImageNet (Shorten & Khoshgoftaar, 2019).(Figure 3.41)

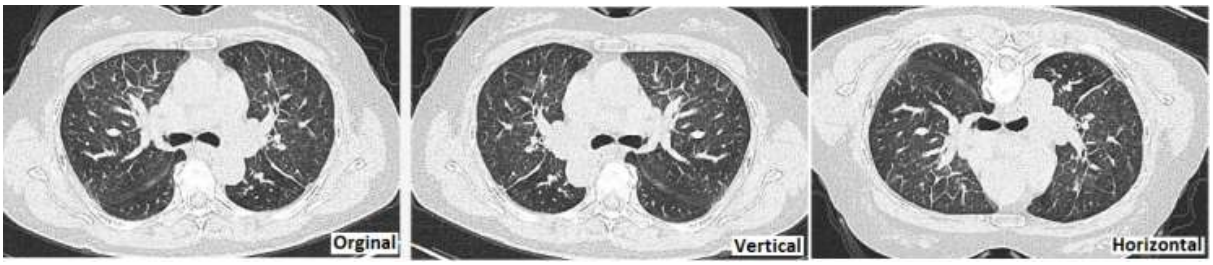

Figure 3.41. Vertical and horizontal flipping

Random erasing: Random erasing is the filling with minimum, maximum, average or a random value of a rectangular region of the image whose position and dimensions are determined randomly(Figure 3.42). With this way, it is aimed to prevent the overfitting of network learning by classifying with obvious features. It has a similar effect to the dropuot

function implemented in a fully connected layer. It ensures a stronger learning by discovering hidden features that are spread over the whole image (Shorten & Khoshgoftaar, 2019).

Figure 3.42. Random erasing

Rotation: Rotation right or left can be useful for data augmentation. The angle of rotation can be determined randomly within a certain range. Studies propose to limit the absolute value of the rotation angle to 20º for the safety of the classification success (Shorten & Khoshgoftaar, 2019).(Figure 3.43)

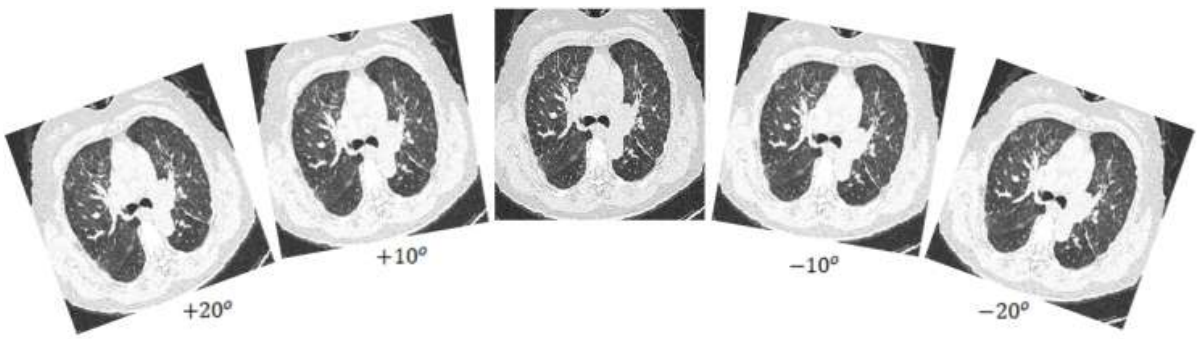

Figure 3.43. Rotation with different angle values

Corners of the image may be cropped in the fixed size rotation process. Changes in image dimensions may be necessary to maintain image entirety.

Translation: The fact that the positions of the distinctive features in the training data are fixed on the image makes this necessary for the test data as well. The way to avoid this dependency is to shift the image along the axes.The image can be shifted to different amounts along different axes to avoid this position dependent learning effect. Gaps caused by shifting can be filled with limit values such as 0 or 255, depending on the encoding format of the image (Shorten & Khoshgoftaar, 2019). Fig.13 shows the images shifted down and filled by 1, (a) right shifted and filled by 0, (b) right and down shifted and filled by 0, (c) left and up shifted and filled by 1 (d) respectively.(Figure 3.44)

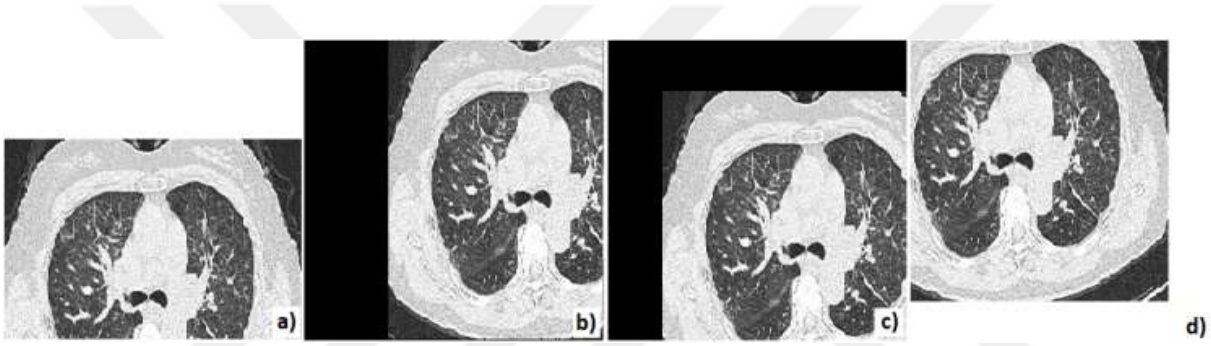

Figure 3.44. Translated and filled (with 0,1) images

Noise Injection: Studies have shown that adding noise to the image gives robustness to the classification success of the network (Zhang et al., 2020) (Shorten & Khoshgoftaar, 2019). Figure 3.45 shows the effect of different noises added to the original image (a).

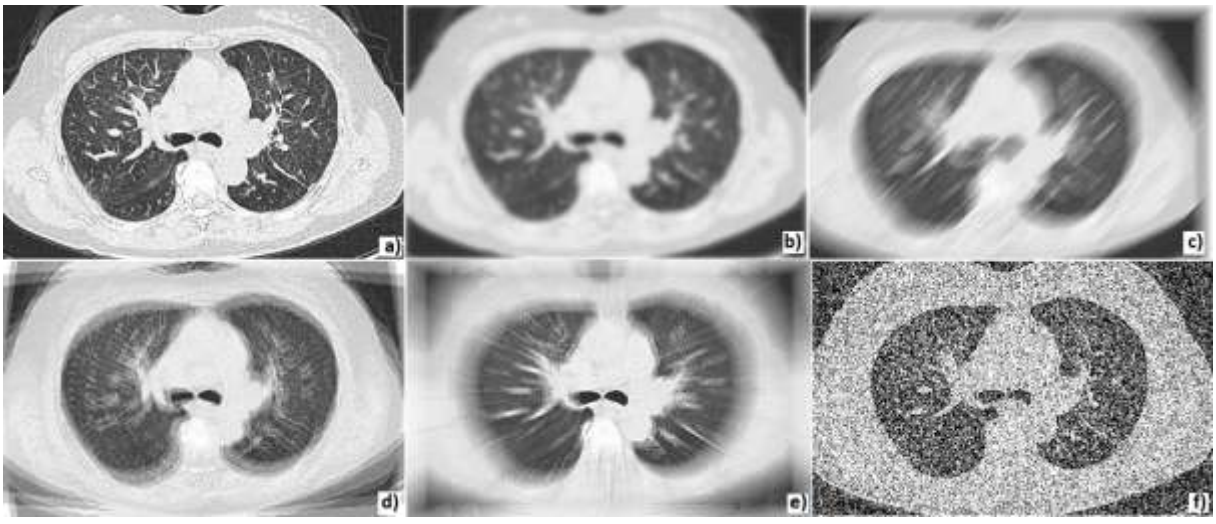

Figure 3.45. Effects of different noise injection

These are Gaussian blur(b), motion blur(c), radial blur(d), zoom blur(e), and salt-peper noise. Gaussian blur is a frequently used noise type in ANN.

3.4.4.2. Color Space Transformations

Color images are expressed with three-dimensional matrices. These are width, length and color channels. Color space transformations are also referred to as photometric transformations.(Shorten & Khoshgoftaar, 2019)

It is useful to highlight a color channel when the color of the image is a distinctive feature. Enriching the appropriate color channel of the image will strengthen the distinctive features. If the distinctive feature consists of a specific color, it is a useful method to produce monochrome images by using this color channels. If the distinctive feature is mostly evident in a color channel, enriching this channel will strengthen the feature.

Biomedical images are created by recording the effects of radioactive magnetic or ultrasonic signals, so they do not contain natural colors. These are usually gray scale coded. Although the biomedical images presented in some sources appear monochrome, the image format may be RGB. These images may be converted to gray level code for size reduction network performance.

3.4.4.3. Kernel Filters

In Figure 3.13, the effect of same filters has been showed. It is possible to increase the size of the data by applying a filter to the image. While the training of the network, CNN filters are also updated to highlight similar features. It may not be an effective method to duplicate data using the same filters (Shorten & Khoshgoftaar, 2019).

3.4.4.4. Mixing Images

Mixing images is a rich method of reproducing data.In this method, images in the existing data set are mixed with various methods and new images are produced. The uncertainty contained in mixed images creates immunity to overfitting.

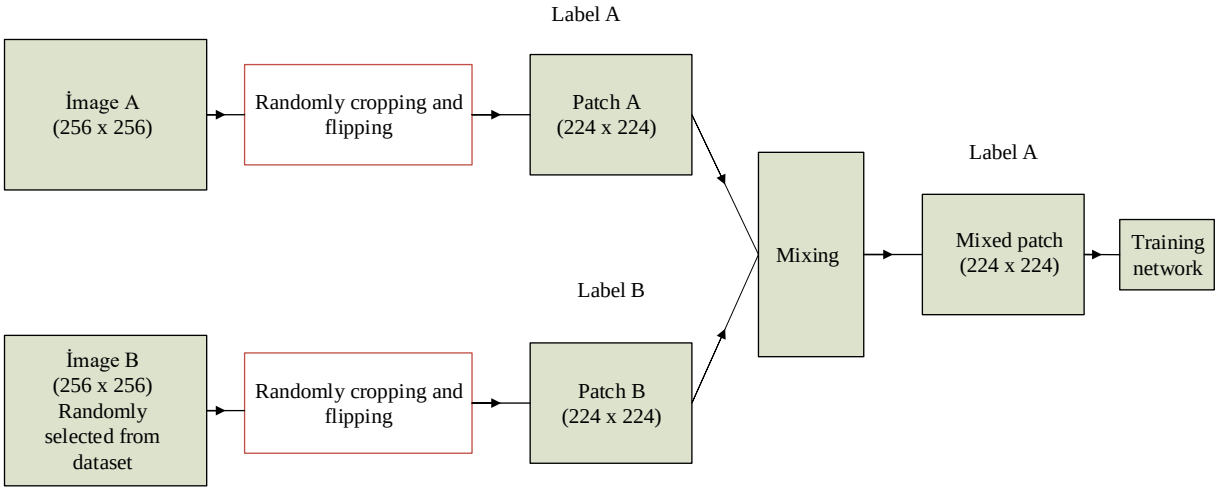

Figure 3.46. Sample pairing augmentation strategy

It has been shown that the new dataset generated with average color values of randomly cropped regions of the two images reduced learning errors. (Shorten & Khoshgoftaar, 2019)

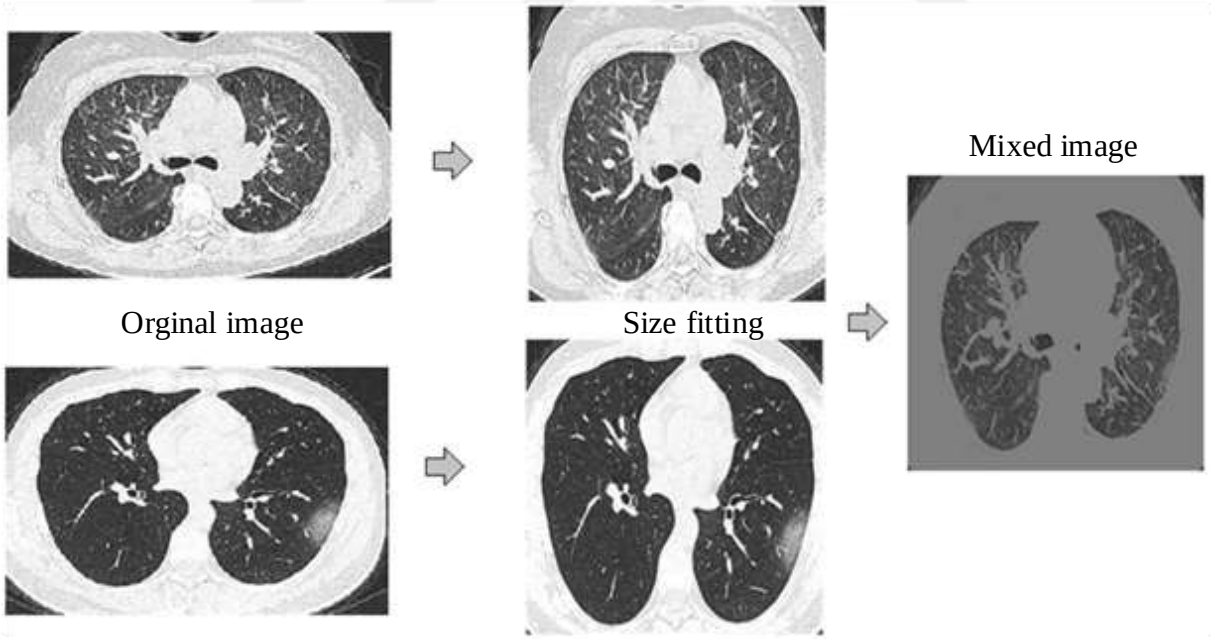

Figure 3.47. Efects of image mixing

It has been stated that this method gives better results with a smaller data set(Inoue, 2018). Figure 3.46 shows the sample pairing strategy and Figure 3.47 shows its effects.

3.5. Deep Learning Frameworks for Constructing CNN

The rapid development and high success of deep learning algorithms have enabled them to find application in real problems in many different fields. The desire for higher achievement has revealed deeper and more complex network architectures. This situation necessitated the separation of architectural and mathematical background in deep learning networks. Different field experts want to create, train, test and use their own models without worrying about the mathematical calculations of deep learning networks. This requirement has given birth to deep learning frameworks. These frameworks offer more than just these conveniences.

One of these conveniences is about the effective use of computer hardware. Deep learning algorithms are basically based on vector operations. Since computer graphics cards contain specialized processors for these processes, they provide great speed increases in the training of deep learning algorithms. Deep learning frameworks allow a supported graphics card to be used in training the network.

Another advantage of these frameworks is that they can run processors or cluster computers in parallel to train networks. In this way, the training time of the networks is shortened. Another advantage of these frameworks is that they offer pre-trained network models. Since the pre-trained network models have been trained with similar data set before, the training parameters are set with a correct initial value. Training speed and network success increase when these models are trained with new datasets.

These fameworks also provide the opportunity to automatically download data sets presented on the internet repository. In this section, some well-known deep learning frameworks are briefly introduced. The success of these frameworks can be evaluated according to speed, extensibility, and hardware utilization and speed criteria(Bahrampour et al., 2015). Extensibility is adaptability to different deep learning architectures. Hardware utilization explains CPU-GPU compatibility, multithreading, and paralleling capabilities. Deep learning frameworks focus on accelerating feed-forward and feedback processes to shorten the training times of networks. In order to get rid of the cyclic coding blocks that negatively affect the algorithm speed, the operations are based on the vectorization of multidimensional data matrices (tensor).

3.5.1. Caffe

Caffe[8] was developed by the Berkeley Artificial Intelligence Research group and is distributed under a BSD 2-Clause license. It was developed in C++ language. It provides GPU support with CUDA(Mocanu, 2008) library. It offers pre-trained models for familiar network architectures.

3.5.2. TensorFlow

Tensorflow[9] is an open source deep learning framework developed by Google and distributed under the Apache 2.0 license. This framework, which also offers CPU-GPU support, is one of the most widely used frameworks today. Developed with Python language, this framework also supports other languages. It has libraries adapted for different development environments (desktop, server, mobile etc.).

3.5.3. Theno

Theno[10] is a symbolic mathematical Python library developed by the Montreal Institute for Learning Algorithms. It is customized for deep learning tensor calculations. It uses (Mocanu, 2008) for GPU support. It is built on top of NumPy library of Python and distributed uder BCD license.

3.5.4. Torch

Torch[11] is a computational framework with powerful machine learning support. Like many other frameworks, it offers GPU support thanks to the CUDA(Mocanu, 2008) library. It is developed in the Lua language and is run with the Lua JIT compiler. Torch7 offers multi-GPU support. Lua community offers powerful open source libraries in many different fields.

3.5.5. Neon

Neon[12] is Intel's reference deep learning framework developed by Nervana System. Supports common networks such as RNN, LSTM, GRU, BatchNorm. It offers a vast ecosystem of pre-

[8] https://caffe.berkeleyvision.org/
[9] https://www.tensorflow.org/
[10] https://pypi.org/project/Theano/
[11] http://torch.ch/
[12] https://github.com/NervanaSystems/neon

trained networks. Improved CPU performance for deep learning calculations by enabling Intel Math Kernel Library (MKL).

3.5.6. MatConvNet

MatConvNet is a CNN library developed by (Vedaldi et al., 2015) for Matlab. The purpose of the development of this library is to enable the CNN application to be developed with Matlab codes in the Matlab program, which has applications in various scientific computing fields. MatConvNet, which also offers GPU support with its CUDA(Mocanu, 2008) library, offers models for many pre-trained networks. The use of the MatConvNet library, in which the experimental application of this thesis is made, is explained in detail in the Section: 3.6.

3.6. Classical CNN Arthitectures

The adventure of creating deeper networks for higher performance, which started with the introduction of LeeNet-5 in 1998, continues. In this section, some architecture, which are the first examples of CNN and deep CNN models, are briefly mentioned.

3.6.1. LeNet-5

LeeNet is a seven-layer CNN model proposed by (Lecun et al., 1998) in 1998. These are two convolutions, two pooling and three fully connected layers. It forms the foundations of modern CNN architectures(Li et al., 2020). Developed for handwritten digit recognition, this network takes 32x32 single-channel input images.

3.6.2. AlexNet

The model proposed by (Krizhevsky et al., 2012) in 2012 is the winner of the ImageNet 2012 competition held in the same year. It consists of five convolutions and three fully connected layers. After each convolution operation, max pooling was applied. The convolution filters used in the layers of the network that receives 224x224x3 input images are 11, 5, 3, 3 and 3 sizes, respectively. It is designed to operate with two GPUs to increase network training speed. ReLU activation function, release, and local response normalization have been proposed for the first time in the CNN model(Li et al., 2020).

3.6.3. VGGNets

Developed by the Oxford University Visual Geometry Group (VGG), there are several versions of this model(Simonyan & Zisserman, 2015). Each version consists of a different number of layers in the range of 11-19, and it was emphasized that the increase in the number of layers increased the success of the network. The convolution filters used in this network are 3x3 in size and are smaller than the filters used in other networks. As a result, the number of parameters of the network is reduced by about 45%.(Li et al., 2020)

3.6.4. GoogLeNet

GoogleNet is a CNN architecture proposed by (Szegedy et al., 2015) for the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). It is the first large-scale network model built using modules known as Inception(Li et al., 2020). Inception modules have four different versions created by the development of the first of these, Inception V1. The V1 version has a block structure formed by placing 1, 3, 5 dimensional filters in parallel or sequentially. While the different sized filters used are effective in detecting different features, the small size of the filter is effective in reducing the computational difficulty caused by the depth of the network.

3.6.5. ResNet

Resnet, the winning model of ILSVRC 2015, was introduced by (He et al., 2016). The model, which gains depth with the addition of two- and three-layer residial blocks, overcomes the vanishing gradient problem with short-circuit connections of the blocks. Since the short-circuit mechanism does not contain any training parameters, it also shortens the training time of the network. To increase the effectiveness of the ResNet model, 34, 50, 101, 150 layer models have been created and tested. Different modifications of the ResNet model have also been developed(Li et al., 2020).

3.6.6. MobileNets

MobileNet was developed by Google for display applications on mobile devices and embedded systems(Howard et al., 2012). These devices have to strike a balance between speed and accuracy in deep learning applications due to the limited hardware they have due to their nature. MobileNet establishes this balance with the "latency" and "accuracy" hyperparameters. MobilNets has three different versions(Li et al., 2020).

3.7. CNN Implementation for Diagnosis of the Pandemic

The application development step is grouped under four headings. These are selection of the coding environment, dataset selection, model selection, and training and testing of the network.

3.7.1. Coding Environment and Framework

Application studies were carried out on a laptop computer with Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz processor, 8GB ram memory capacity, 64 bit Windows 10 operating system installed.

The model was created using the MatConvNet(Vedaldi et al., 2015) library in the Matlab R2018b program. The starting point is the "custom_imdb" work in this library. The "cnn_toy_data_generator.m" file here has been modified to convert the image files on the local drive to imdb format. The file "cnn_toy_data.m" has been modified to create the model and specify the training parameters. In order to test the created model, the function that evaluates the network object and test images and presents the results is coded in the "testrun.m" file.

3.7.2. Dataset preparation

The "SARS-CoV CT-scan Dataset"(Soares et al., 2020) and COVID-CT-Dataset (Yang et al., 2020) datasets was used in experimental studies. Datasets are divided into three groups as training, validation and test data. For first dataset, there are 1024 Covid-positive and 1024 Covid-negative images in the training set. The both validation and testing datasets consists of 100 Covid-positive and 100 Covid-negative diagnosed data. For secod dataset traninig dataset consist of 180 Covid-positive and Covid-negative data. The both validation and testing datasets consists of 60 Covid-positive and 60 Covid-negative diagnosed data.

Since the data set contains consecutive slices of lung the CT images of the same person, the following method was used in the selection of validation and test data for first dataset. Images with a sequence number multiple of 10 were selected for validation data. Among the remaining images, the test data was selected with the same method.

Since the second data set usually consists of independent images, the following method was used in data selection. To equalize the number of positive and negative cases in the dataset, images of low quality and size were eliminated. Among the remaining data, validation and test data were selected in equal numbers and sequentially from the large, medium and small groups according to the file size.

The dataset preprocessing used in the studies with the most successful model, Model-8, is different from the others. The dataset used is Soares' dataset (Soares et al., 2020). Here, validation and test data are the same and consist of 100 images randomly selected from the original data. The training data is 1129. Data augmentation methods are rotation and translation. 2 horizontal and 2 vertical translations were applied to both groups. Scroll amount is +10px and -10px. The amount of rotation is 5 and 15 degrees to the right and left. In the last case, the number of training data is 10161 and the number of test data is 900.

Images are adjusted to the dimensions specified for each application model. Data is in single precision format with one channel. Normalization was applied by subtracting the average of all data from each image.

3.7.3. Creating CNN Models

The models are formed by adding convolution, pooling and activation layers sequentially in the MatConvNet framework. The softmax loss function is applied in the last layer.

Because convolution and pooling operations change the output image size, the filter, stride, and padding values are adjusted to be consistent with the model. These adjustments were made using Equations: 17, 18, 19.

The filter values used in the convolution are one of the basic parameters of the model. A balance must be struck between the classification accuracy of the network and the network performance when determining the size and number of filters. Filter sizes are related to the spatial size of the features to be detected in the image. The number of filters, on the other hand, increases the depth of the network and enables learning of complex abstract features. Really input and output sizes are four dimensional tensors. For simplify third and fourth channel was omitted in tables. For example in the last layer, the input data size is [1x1x1x2].

Here 2 is the number of classes in the classification layer. The models created for experimental studies are below.

Table 3.3. Arthitecture of CNN Model-1

Model No	Layer No	Layer Type	Input Size	Filter Size	Stride size	Padding size	Filter Count	Output size
1	1	Conv	32x32	9x9	1	0	15	24x24
	2	Maxpool	24x24	2x2	2	0	-	12x12
	3	Conv	12x12	7x7	1	0	5	6x6
	4	Maxpool	6x6	2x2	2	0	-	3x3
	5	Conv	3x3	3x3	1	0	2	1x1
	6	Softmaxloss	1x1	-	-	-	2	1x2

The difference between Model-1 and Model-2 is the ReLU activation functions applied after the two maxpool layers.

Table 3.4. Arthitecture of CNN Model-2

Model No	Layer No	Layer Type	Input Size	Filter Size	Stride size	Padding size	Filter Count	Output size
2	1	Conv	32x32	9x9	1	0	15	24x24
	2	Maxpool	24x24	2x2	2	0	-	12x12
	3	ReLU	12x12	-	-	-	-	12x12
	4	Conv	12x12	7x7	1	0	5	6x6
	5	Maxpool	6x6	2x2	2	0	-	3x3
	6	ReLU	3x3	-	-	-	-	3x3
	7	Conv	3x3	3x3	1	0	2	1x1
	8	Softmaxloss	1x1	-	-	-	2	1x2

The difference of Model-3 from Model-2 is the 7^{th} layer, the dropout layer. The only variable of this layer is the dropout rate

Table 3.5. Arthitecture of CNN Model-3

Model No	Layer No	Layer Type	İnput Size	Filter Size	Stride size	Padding size	Filter Count	Output size
3	1	Conv	32x32	9x9	1	0	15	24x24
	2	Maxpool	24x24	2x2	2	0	-	12x12
	3	ReLU	12x12	-	-	-	-	12x12
	4	Conv	12x12	7x7	1	0	5	6x6
	5	Maxpool	6x6	2x2	2	0	-	3x3
	6	ReLU	3x3	-	-	-	-	3x3
	7	Dropout	3x3	Dropout rate=0.4 - 0.5				3x3
	8	Conv	3x3	3x3	1	0	2	1x1
	9	Softmaxloss	1x1	-	-	-	2	1x2

. This layer does not contain learnable or size-reducing parameters. Dropout aims to spread learning across the network by closing some of the paths that the network memorizes.

Table 3.6. Arthitecture of CNN Model-4

Model No	Layer No	Layer Type	İnput Size	Filter Size	Stride size	Padding size	Filter Count	Output size
4	1	Conv	32x32	9x9	1	0	45	24x24
	2	Maxpool	24x24	2x2	2	0	-	12x12
	3	ReLU	12x12	-	-	-	-	12x12
	4	Conv	12x12	7x7	1	0	15	6x6
	5	Maxpool	6x6	2x2	2	0	-	3x3
	6	ReLU	3x3	-	-	-	-	3x3
	7	Dropout	3x3	Dropout rate=0.4 and 0.6				3x3
	8	Conv	3x3	3x3	1	0	2	1x1
	9	Softmaxloss	1x1	-	-	-	2	1x2

The dropout rate should be adjusted appropriately as it provides a balance of learned information and memorized information. The difference between Model-4 and Model-3 is the number of filters in the convolution layers. The weights in the classical NN are replaced by filters in CNN. Both the number of filters and the filter sizes are parameters that affect the performance of the network. Since the convolution process and back propagation are applied to all filters, these filter parameters should be adjusted optimally. Depth increase is achieved with added layers. The difference of Model-5 from Model-4 is its first layer.

Table 3.7. Arthitecture of CNN Model-5

Model No	Layer No	Layer Type	İnput Size	Filter Size	Stride size	Padding size	Filter Count	Output size
	1	Conv	64x64	33x33	1	0	32	32x32
	2	ReLU	32x32		-	-	-	32x32
5	3	Conv	32x32	9x9	1	0	16	24x24
	4	Maxpool	24x24	2x2	2	0	-	12x12
	5	ReLU	12x12	-	-	-	-	12x12
	6	Conv	12x12	7x7	1	0	8	6x6
	7	Maxpool	6x6	2x2	2	0	-	3x3
	8	ReLU	3x3	-	-	-	-	3x3
	9	Dropout	3x3	Dropout rate=0.4				3x3
	10	Conv	3x3	3x3	1	0	2	1x1
	11	Softmaxloss	1x1	-	-	-	-	1x2

Convolution and pooling parameters in this layer adapt the network to 64x64 images. Testing with the Model-5 has shown that training this model with the current hardware and augmented data will take a lot of time. This is because of the filter sizes and the number of filters in the first convolution layer. Model-6 was developed to reduce the training cost of the model. In the new model, the size and number of filters in the first layer have been reduced.

Table 3.8. Arthitecture of CNN Model-6

Model No	Layer No	Layer Type	İnput Size	Filter Size	Stride size	Padding size	Filter Count	Output size
6	1	Conv	64x64	7x7	3	0	15	20x20
	2	Maxpool	20x20	2x2	2	0	-	10x10
	2	ReLU	10x10	-	-	-	-	10x10
	3	Conv	10x10	5x5	1	0	20	6x6
	4	Maxpool	6x6	2x2	2	0	-	3x3
	5	ReLU	3x3	-	-	-	-	3x3
	6	Conv	3x3	3x3	1	0	10	1x1
	8	ReLU	1x1	-	-	-	-	1x1
	9	Dropout	3x3	Dropout rate=0.6				3x3
	10	Conv	1x1	1x1	1	0	2	1x1
	11	Softmaxloss	1x1	-	-	-	-	1x2

The number of filters in the middle layer has been increased. Since the input image sizes are reduced in the middle layer, this will reduce the negative effect of increasing the number of filters on the training speed.

As the input image sizes increase, the defining texture and edge features of the image become more pronounced. However, this increases the training time by increasing the number of training parameters of the network.

The Model-7 was developed to use larger images on our network. In the first layer of this model, 9x9 filter and stride=3 values are used. Initially 100x100, the image size is 32x32 at the output of the first convolution layer and has been reduced to a very reasonable size for subsequent layers.

Table 3.9. Arthitecture of CNN Model-7

Model No	Layer No	Layer Type	İnput Size	Filter Size	Stride size	Padding size	Filter Count	Output size
7	1	Conv	100x100	9x9	3	1	15	32x32
	2	Maxpool	32x32	2x2	2	0	-	16x16
	3	ReLU	16x16	-	-	-	-	16x16
	4	Conv	16x16	5x5	3	2	15	6x6
	5	Maxpool	6x6	2x2	2	0	-	3x3
	6	ReLU	3x3	-	-	-	-	3x3
	7	Dropout	3x3	Dropout rate=0.4				3x3
	8	Conv	3x3	3x3	1	0	2	1x1
	9	Softmaxloss	1x1	-	-	-	2	1x2

Model-8 is a model developed as a solution to the learning problems in previous models.

Table 3.10 Arthitecture of CNN Model-8

Model No	Layer No	Layer Type	İnput Size	Filter Size	Stride size	Padding size	Filter Count	Output size
8	1	Conv	244x244	11x11	11	10	35	24x24
	2	ReLU	24x24	-	-	-	-	24x24
	3	Conv	24x24	7x7	3	2	45	8x8
	4	ReLU	8x8	-	-	-	-	8x8
	5	Conv	8x8	5x5	1	0	60	4x4
	6	ReLU	4x4	-	-	-	-	4x4
	7	Conv	4x4	3x3	1	0	120	2x2
	8	ReLU	2x2	-	-	-	-	2x2
	9	Dropout	2x2	Dropout rates: 0.1 – 0.9				2x2
	10	Conv	2x2	2x2	1	0	2	1x1
	11	Softmaxloss	1x1	-	-	-	2	1x1

In this model, the input image is 244x244. In this model, unlike other models, no dropout layer has been added. Filter sizes were started with 11x11 and gradually reduced. The number of filters increases in the following layers. The loss of properties caused by the reduction in size as a result of convolution is compensated by increasing the number of filters. It is aimed to prevent feature losses that will occur by removing the pooling layer. In the experimental studies, it was seen that the most successful model was the Model-8.

3.7.4. Creating Fuzzy Models

When a trained network run to test a specific sample the value produced should be interpreted together with the statistical success of the network. For a spesific example, a network that generates a high rate of diagnostics may have poor statisticial classification success. In this case, the reliability of the classification success of the single sample will also be low. It is possible to interpret the classification success with a fuzzy model which is applied to the CNN output.

In this section, two single-input-single-output Mamdani Fuzzy Logic model was applied to improve the classification efficiency of the CNN-based medical image classification problems(Figure 3.48) The CNN output is produced with binary softmax function and these outputs are used as the input for created fuzzy models.

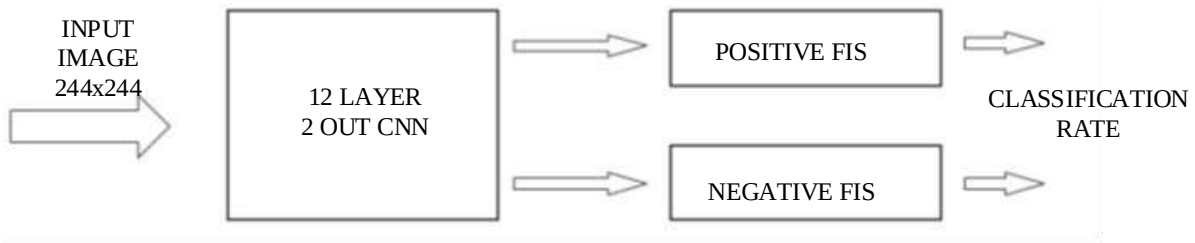

Figure 3.48. Buildig block of fuzzy sysetms with CNN

In the selection of the membership functions of the models, the histogram graph and confusion matrix of the test data is utilized. The only output of the model represents the interpreted classification success. The confusion matrix for the test data is presented in Table 3.11. In the table, it is seen that the correct classification rate of the network is higher for positive diagnoses.

Table 3.11. Confusion matrix of test data

	YES	NO	SUCCES RATE
YES	TP= 895	FP= 5	0.994
NO	FN= 66	TN= 834	0.926

These results are also seen in the histogram graph of the test data presented in Figure 3.49.

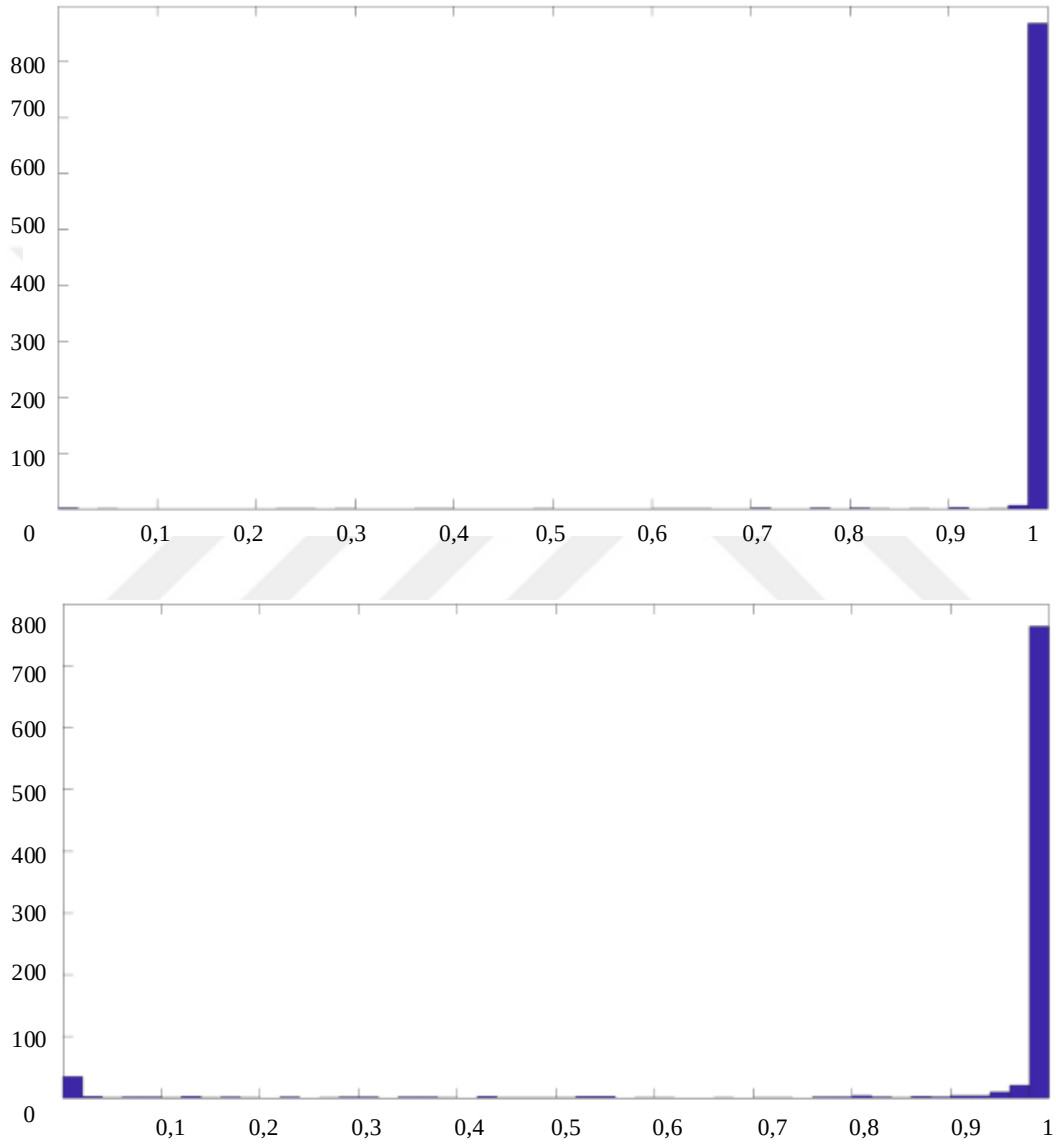

Figure 3.49. The CNN output histogram plot for positive (top) and negative(bottom) diagnostic classes of test data

It is expected that the model output will produce a higher value in positive diagnostics data for same input. This is provided by the input-output membership functions shown in Figure 3.50.

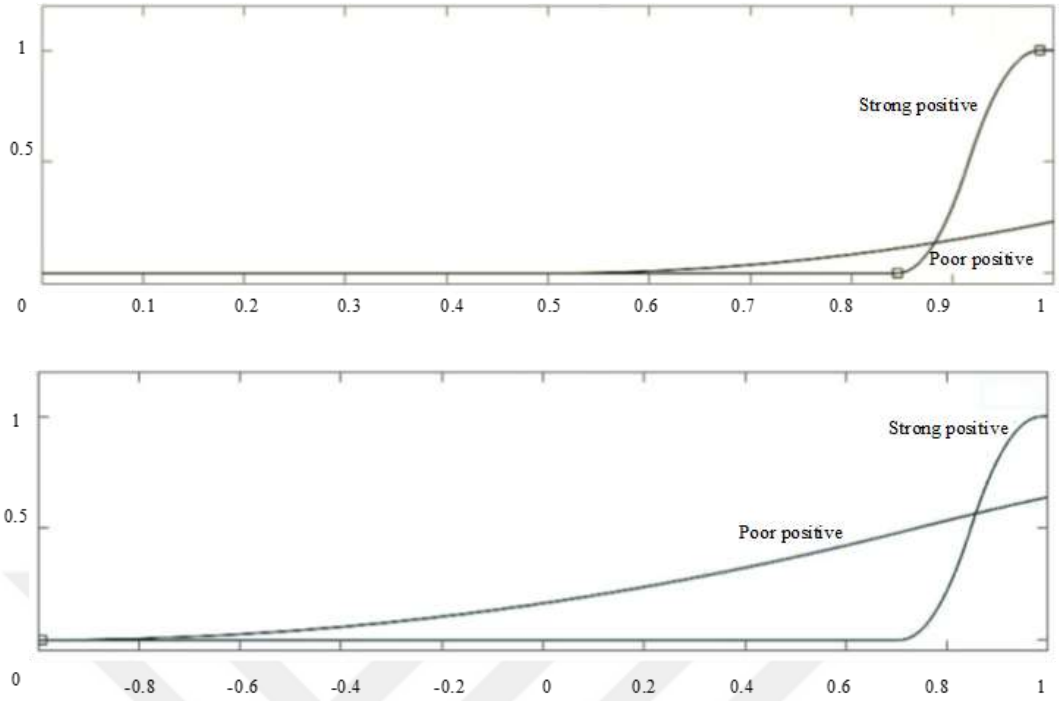

Figure 3.50. Input-Output membership function for samples with positive diagnosis.

Since values less than 0.5 indicate misclassification, the classification success should be smaller for negative diagnoses scattered over the entire range(Figure 3.49).

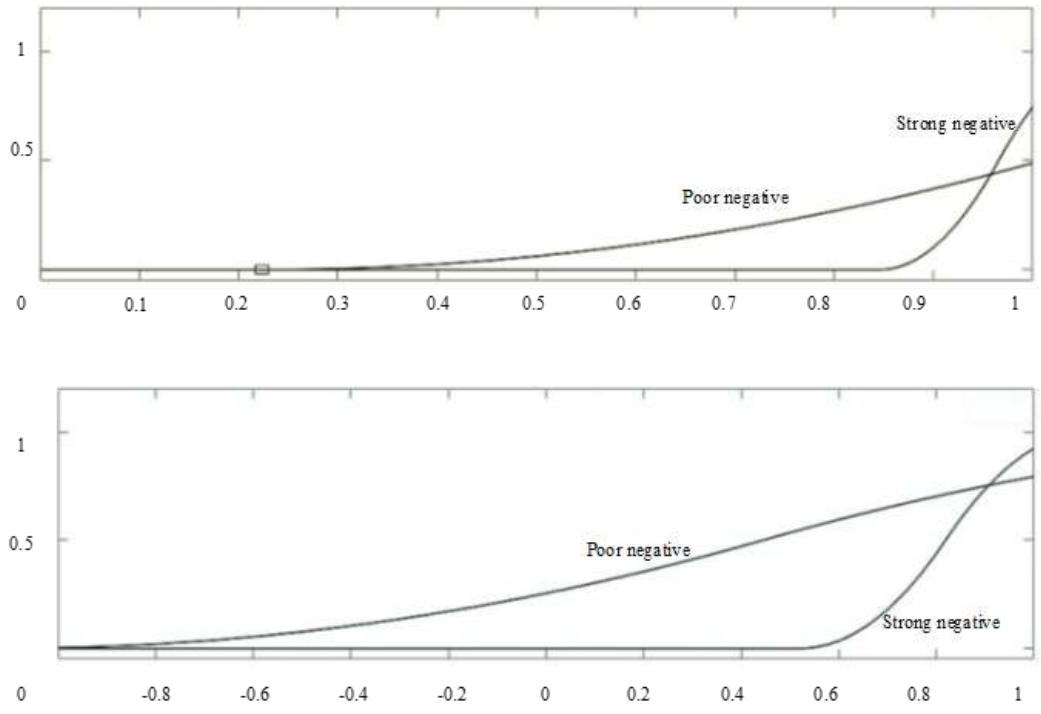

Figure 3.51. Input-Output membership function for samples with negative diagnosis.

This is provided by the input-output membership functions shown in Figure 3.51. Fuzzy models were made with Matlab Fuzzy Logic Toolbox. When the first output of the CNN is bigger (positive diagnosis), the commented output is produced by the first model. Otherwise, the second model works(Figure 3.48). The function triggered at output that similar to the function activated for the input. Output is different from the input due to different parameters.

Table 3.12. Parameters of S-shape functions of fuzzy models

	INPUT		OUTPUT	
	POOR	STRONG	POOR	STRONG
POSITIVE MODEL	[0.4985 1.968]	[0.8452 0.986]	[-0.9921 2.47]	[0.7063 0.988]
NEGATIVE MODEL	[0.8457 1.086]	[0.8457 1.086]	[-1.109 2]	[0.5214 1.117]

The thick vertical line at the output is the defuzzification point(Figure 3.52). FIS membership function parameters are set manually by shifting the graph(Table 3.12).

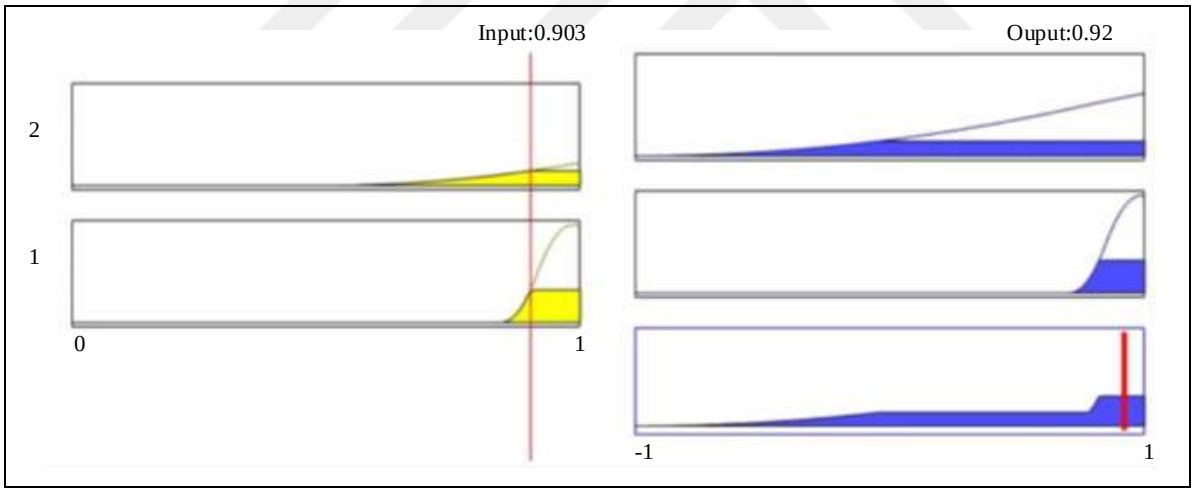

Figure 3.52. Input, output and defuzzification sample for fuzzy models.

Two membership functions are defined for the inputs and outputs. These represent strong and weak diagnoses. The FIS rule set is simply as in Figure 21. It is clear that the strong input triggeres strong output and weak input triggers weak output. The positive and negative fuzzy models are similar.

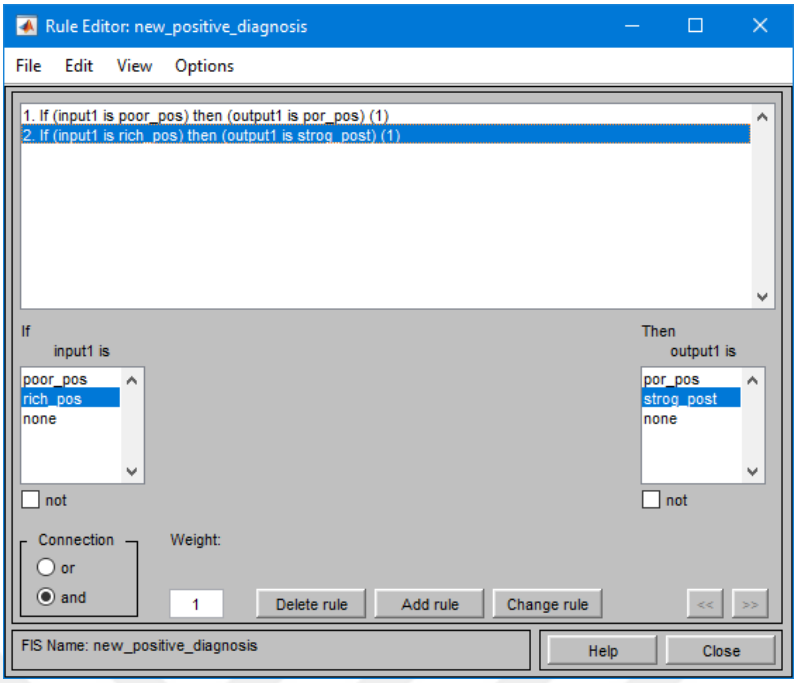

Figure 3.53. Rule editor for positive fuzzy model

The middle of maximums (MOM) is used as the defuzzification function. Other parameters of the model are shown in (Table 4).

Fuzzy Model Type	Mamdani
And Method	Minimum
Or Method	Maximum
Implication	Minimum
Aggregation	Maximum

The algorithmic order of the all operations done so far is as follows.

1. *Prepare dataset*
2. *Create CNN model*
3. *Train and test model*
4. *Evaluate the test results(confusion matrix)*
5. *Create fuzzy model consistent with the test results.*
6. *Test a specific image with trained CNN.*
7. *Apply the CNN output as input to the fuzzy model.*
8. *Get the interpreted classification result.*

The use of fuzzy system contributes to the classifier success in multi-classification task(Tosunoğlu, 2009). However, such a study is beyond the scope of this paper.

4. RESULTS AND DISCUSSIONS

The CNN models created in Section 3.6.3 are trained with different training parameters. Training results are presented in tables and graphics. The results obtained by the evaluation of the tables were interpreted comparatively.

4.1. Training Results

Training experiments are reported in subheadings to evaluate the effects of different parameters on the performance of CNN. While the parameter whose effect was measured in each title was changed, other parameters were kept constant.

4.1.1. Effects of batch size

The data obtained as a result of training the CNN model presented in Table 3.3 with the training parameters in Table 4.1 are also presented in the relevant column of Table 4.1.

Table 4.1 Trainin parameter and results for Train 1-6

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
1	1	32	0,002	100	290	0.755	0.729
2	1	32	0,002	200	578	0.800	0.791
3	1	32	0,002	300	865	0.855	0.798
4	1	64	0,002	100	317	0.680	0.686
5	1	64	0,002	200	601	0.820	0.810
6	1	64	0,002	300	871	0.835	0.835

When Train-1 and Train-4 are compared, it is seen that there is not a big difference between the training times. This may be due to the fact that both models are large enough not to cause memory problems for the hardware. On the other hand, for all epoch values, the training time for the large batch value is high. Another finding is that there is no significant difference in accuracy and F1-score value for two different batch sizes.

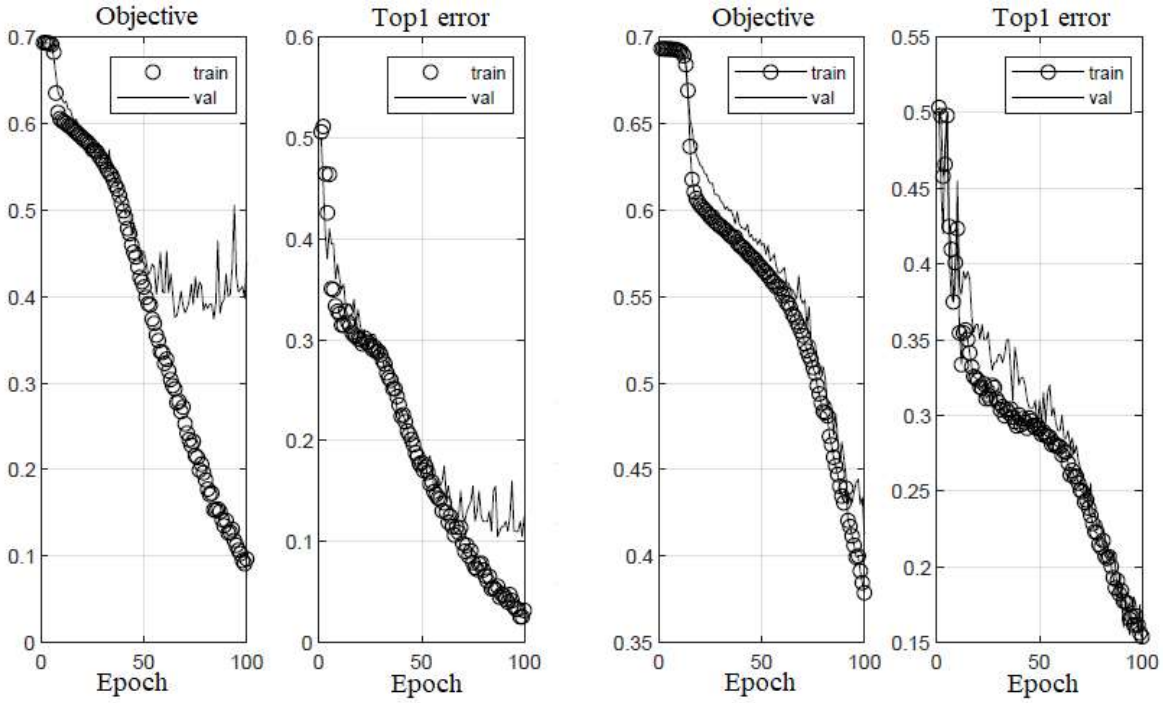

Figure 4.1. First 100 itrations for Train-1 and Train-4

In Figure 4.2, while the train values for batch size=32 became zero in the 150th iteration, they did not reach zero for batch size=64 yet. The reason for this is that the average error is smaller in large batch values. As a result, the update rate of filters decreases during backpropagation.

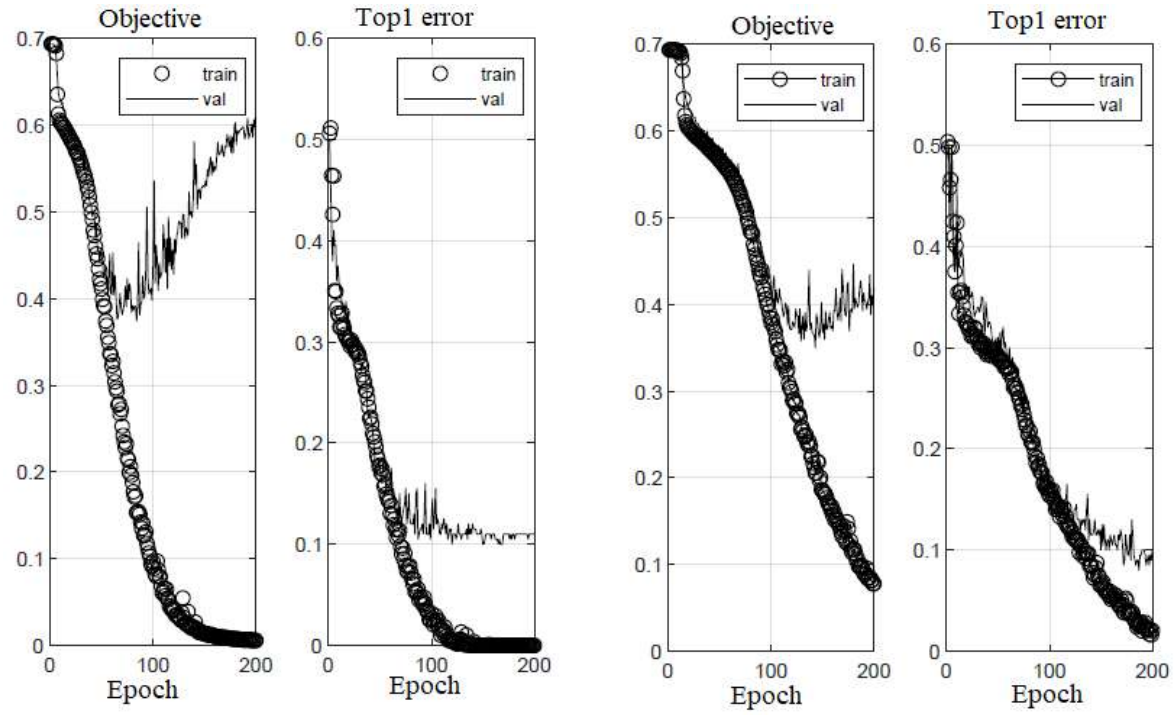

Figure 4.2. First 200 itrations for Train-2 and Train-5

In figure 4.3, even though the graphs are flattened between 200 and 300 iterations for epoch size=32, the error value test success continues to increase. Although the train values for epoch size=64 are zero in Figure 4.3, the test success is behind epoch size=32. On the other hand, the F1-scor value is higher for epoch size=64.

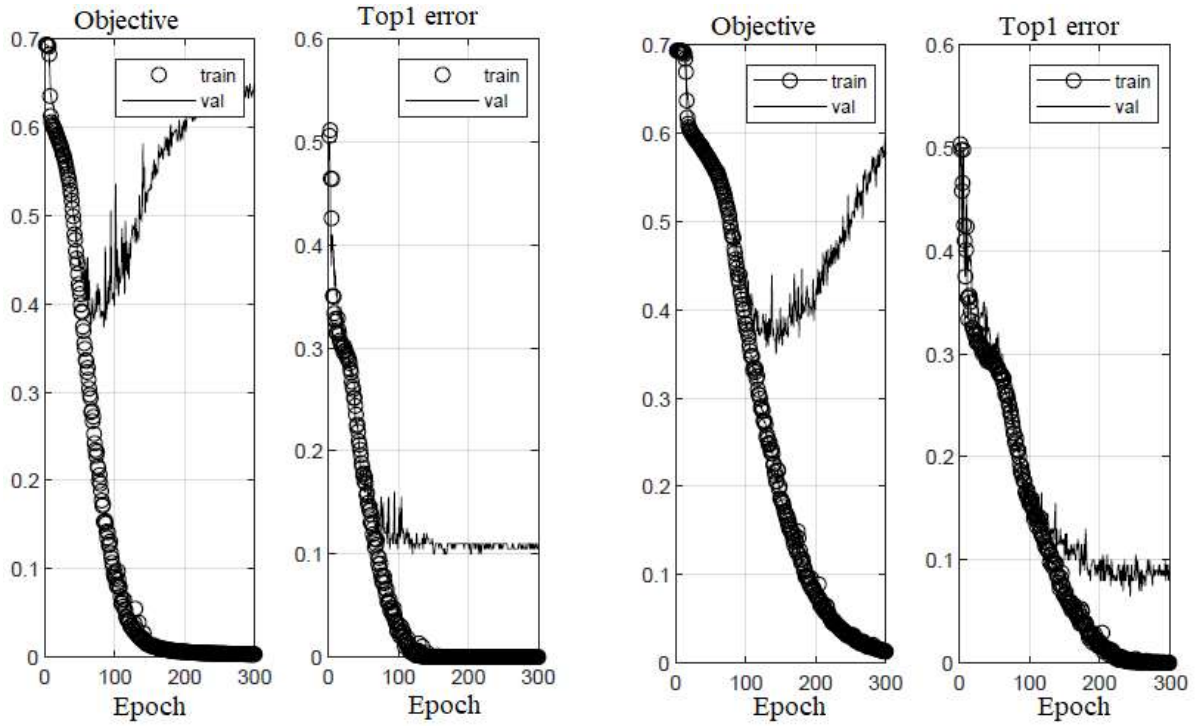

Figure 4.3. First 300 itrations for Train-3 and Train-6

The effect of batch size on the training speed of the network is commented in Section 4.1.7.

4.1.2. Effects of activation function

ReLU activation functions are added after the pooling layers of the model given in Table 3.3. The new model created is shown in Table 3.4. The effects of the ReLU activation function were examined with the training results in Table 4.1 and Table 4.2. When Table 4.1 and Table 4.2 are compared, ReLU activation function for both 32 and 64 batch values increased the classification success and F1-score value. When Figure 4.1 and Figure 4.4 are compared, it is seen that Objective and Top 1 error values are similar. The same is true for Figure 4.2 and Figure 4.5. It is noteworthy in Table 4.1and Table 4.2 that the F1-score value for batc size=64 is closer to the accuracy value than for batc size=32.

Table 4.2. Trainin parameter and results for Train 7-14

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
7	2	32	0,002	100	296	0.895	0.899
8	2	32	0,002	200	587	0.915	0.914
9	2	32	0,002	300	883	0.925	0.925
10	2	64	0,002	100	273	0.835	0.830
11	2	64	0,002	200	549	0.870	0.866
12	2	64	0,002	300	826	0.880	0.879
13	2	64	0,002	400	1108	0.870	0.871
14	2	32	0,002	400	1198	0.925	0.926

In addition, the number of iterations for batch size=64 was increased to 400 and the limits of classification success were observed.

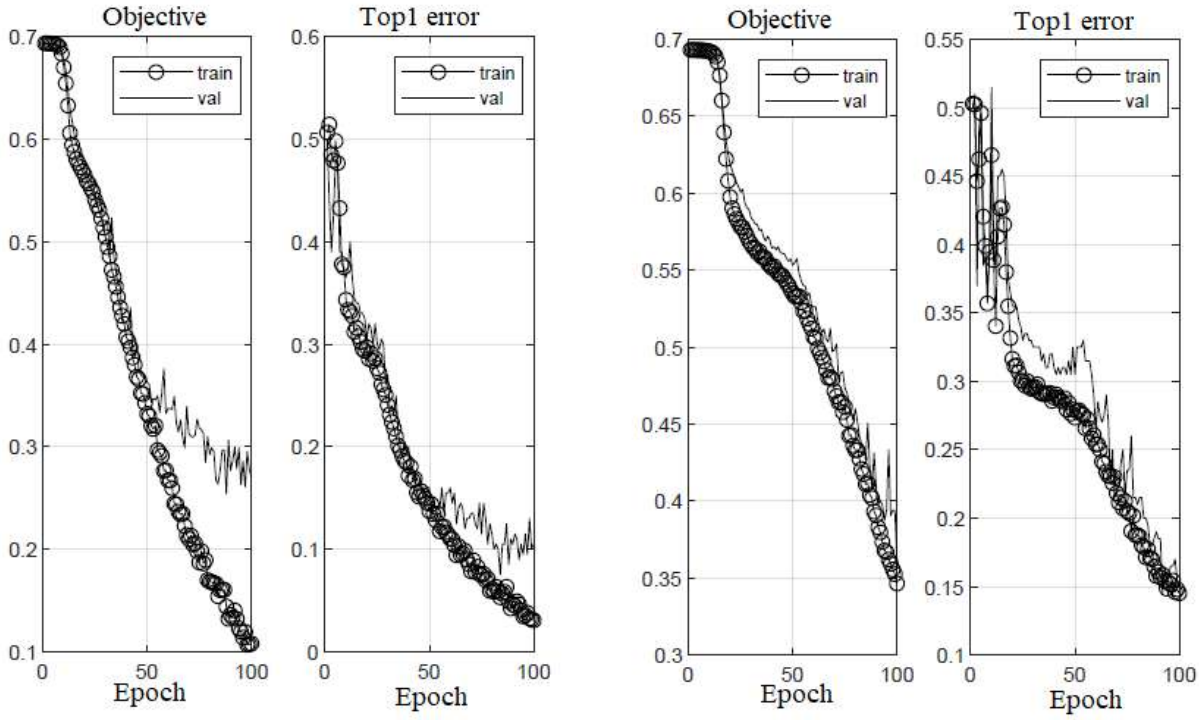

Figure 4.4. First 100 itrations for Train-7 and Train-10

One of the important points in Table 4.2 is that there is a small decrease in classification success after 300 iterations for batch=64.

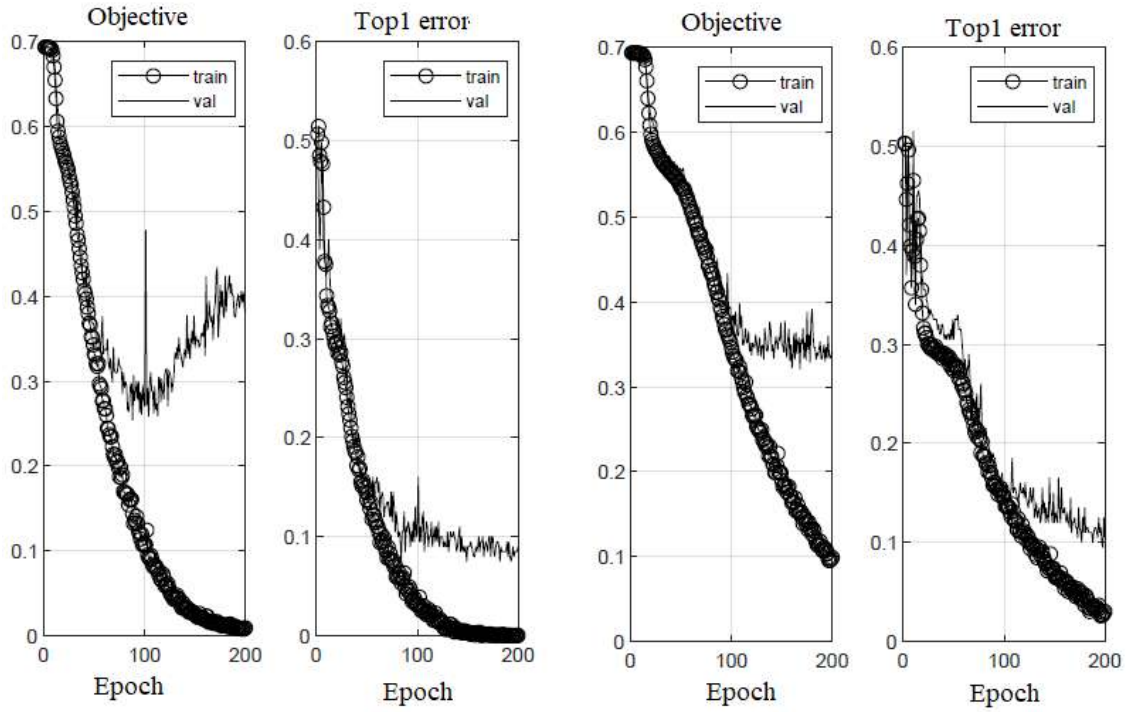

Figure 4.5. First 200 itrations for Train-8 and Train-11

It is seen that there is no change for batch=32 value. It is seen that increasing the number of iterations after this value has no effect on the classification success.

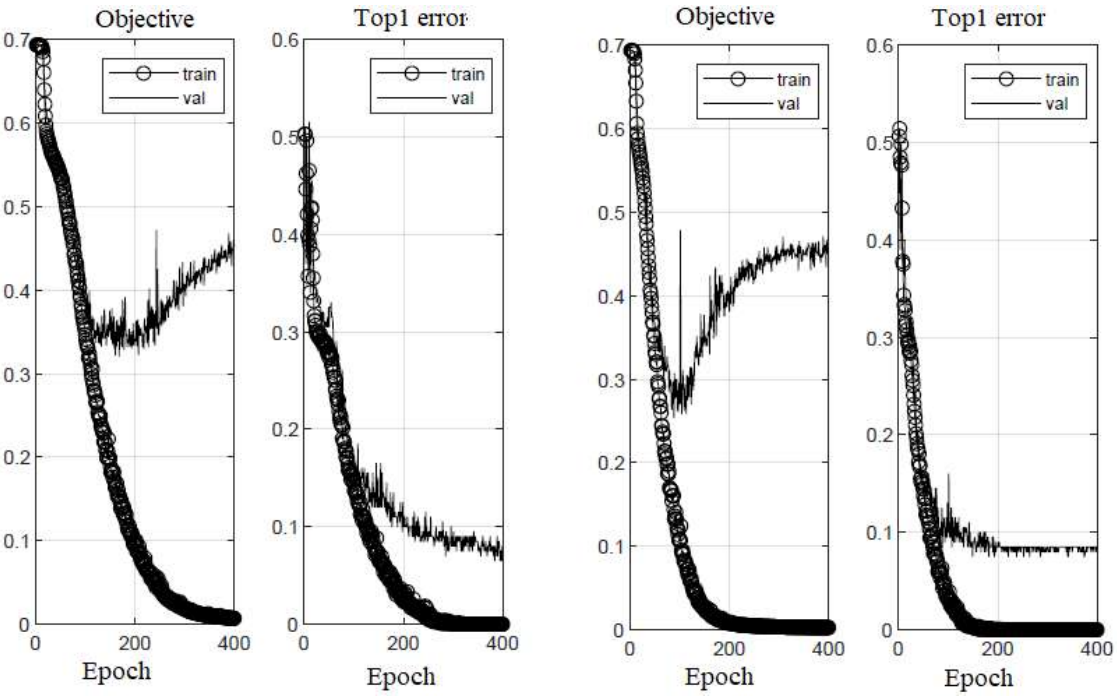

Figure 4.6. First 400 itrations for Train-13 and Train-14

An interesting finding is that for batch=64, although the Top-1 validation value continued to decrease after the 300th iteration, the classification success decreased. (Figure 4.6 and Table 4.2)

4.1.3. Effects of Dropout

All graphs in Section 4.1.1 and Section 4.1.2 show that after certain iteration the training and validation lines are clearly separated from each other. This was an indication that the model memorized instead of learning. This condition is known as overfitting. Dropout is one of the ways to prevent overfitting.

Table 4.3. Trainin parameter and results for Train 15-24

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
15	3	32	0,002	100	360	0.750	0.742
16	3	32	0,002	200	648	0.865	0.868
17	3	32	0,002	300	941	0.850	0.853
18	3	32	0,002	400	1236	0.865	0.874
19	3	32	0,002	500	1534	0.905	0.907
20	3	32	0,002	600	1831	0.900	0.902
21	3	64	0,002	100	276	0.825	0.832
22	3	64	0,002	200	578	0.870	0.876
23	3	64	0,002	300	908	0.935	0.936
24	3	64	0,002	400	1171	0.915	0.918

In this section, the performance of the network is evaluated by adding doropout layers with different dropout ratios to the previously created models. When Figure 4.7 is compared with Figure 4.3 and Figure 4.6, it is seen that the dropout application reduces the training error for the objective function below 0.3. With the interpretation of the figures, it is clearly seen that the epoch value of the point where the train and validation lines diverge has shifted from 100 to 200.

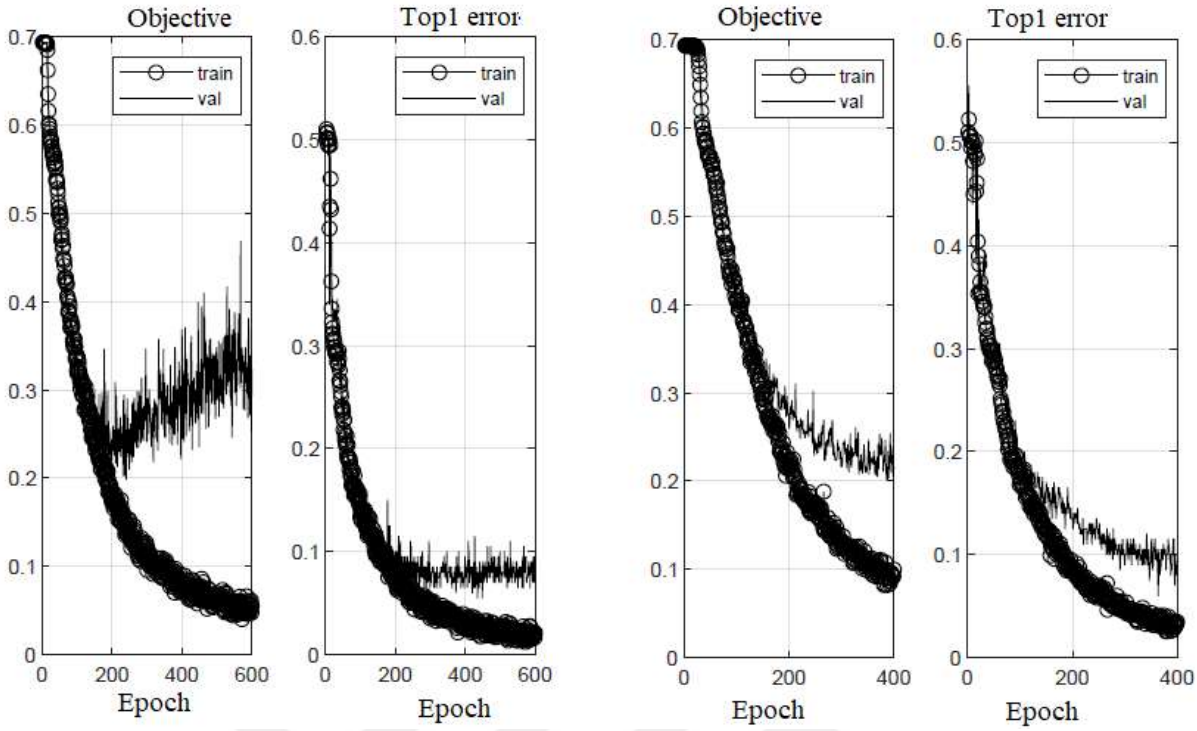

Figure 4.7. First 400 itrations for Train-20 and Train-24

For batch=32, the epoc value required to achieve the highest classification success increased from 300 to 500. Accordingly, it can be said that the dropout application increases the number of iterations for the small batch value. From the comparison of Table 4.2 and Table 4.3, it can be see that the dropout application increases the success of the network by 0.01. For the same iteration number, it is seen that high dropout rate increases the training time of the network in Table 4.5. Here dropout=0.4 for the first four rows in the table, 0.6 for the last four rows.

4.1.4. Effects of Learning Rate

The learning rate is related to the training rate of the network.

Table 4.4. Trainin parameter and results for Train 25-28

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
25	3	64	0,001	100	222	0.550	0.470
26	3	64	0,001	400	1019	0.858	0.859
27	3	64	0,01	100	271	0.900	0.900
28	3	64	0,01	200	531	0.900	0.907

A large learning rate can provide faster learning, while a small learning rate can give more precise results.

When Table 4.4 is examined, the effect of learning rate on training speed is clearly seen. The value reached in 100 iterations with a learning rate of 0.1 could not be reached in 400 iterations with a learning rate of 0.001.

From the interpretation of Table 4.3 and Table 4.4, it is seen that the learning rate does not contribute to the classification success of the network. On the other hand, it is seen that the iteration of the network to reach the best value is lower for big learning rate. Accordingly, the learning rate increased the training speed of the network. The same result is also seen in Figure 4.8

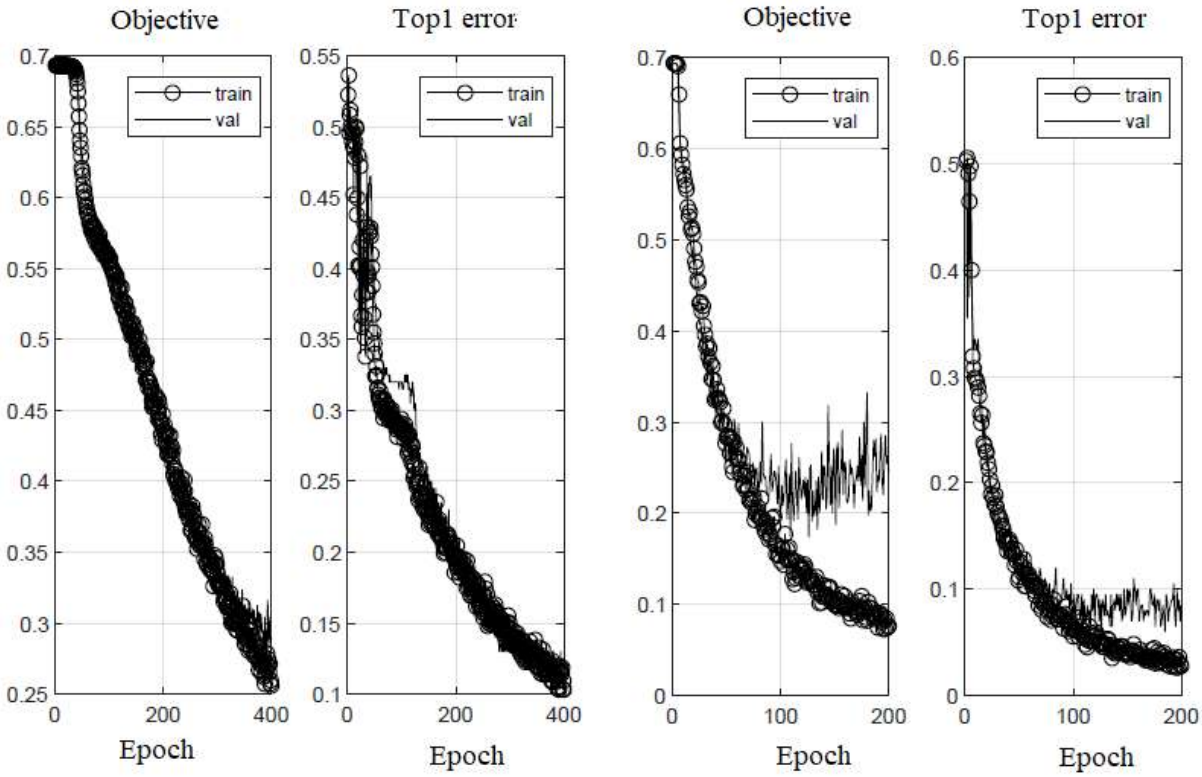

Figure 4.8. First 400 and 200 itrations for Train-26 and Train-28

4.1.5. Effects of Filter Count

The number of filters and their distribution into layers are important for CNN performance. The small number of filters prevents adequate learning of image features. The large number of

filters will increase the number of training parameters and increase the training cost of the network.

In this section, CNN Model-4 (Table 3.6) is developed by tripling the number of filters in the first two convolution layers of CNN Model-3 to experience this effect. The application results are summarized in Table 4.5.

Table 4.5. Trainin parameter and results for Train 29-36

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
29	4	32	0,005	100	549	0.875	0.881
30	4	32	0,005	200	1077	0.910	0.913
31	4	32	0,005	300	1626	0.935	0.934
32	4	32	0,005	400	2243	0.915	0.911
33	4	32	0,005	100	555	0.875	0.881
34	4	32	0,005	200	1055	0.890	0.888
35	4	32	0,005	300	1602	0.925	0.923
36	4	32	0,005	400	2119	0.925	0.927

From the interpretation of Table 4.4 and Table 4.5, it is seen that the training time has doubled. On the other hand, the increase in classification success is about 0.02. In this case, the gain provided by the increase in the number of filters should be evaluated together with the cost it brings. Dropout=0.4 for the first four rows in the table, 0.6 for the last four rows in Table 4.5.

4.1.6. Effects of Data Augmentation

The first dataset(Soares et al., 2020) was used in the experimental studies up to this point. Satisfactory results were obtained in the tests performed with this data set. However, when the trained models are tested with an external dataset (Yang et al., 2020), the classification success is very low. The reason for this was evaluated as overfitting. Because the images in the first dataset contain homogeneous groups consisting of different chest the CT slices belonging to the same person. Different the CT slices of the same person contain similar global attributes. These common features in the test data were evaluated as the source of

overfitting.In this section, the second dataset(Yang et al., 2020) is included in the experimental studies.

Data augmentation is one method of preventing overfitting. In this section, two different data augmentation methods were applied to the data set. In the first method, the image is shifted twice in both directions along the horizontal and vertical axis. Thus, the number of images has been increased 16 times. In the second method, the image was rotated two times in the negative and positive directions with 5 and 10 degrees. With this method, the image is increased 4 times. Augmented images are used together with the original images. The total number of data is twenty-one times the number of images. The method was applied to training and validation and test data. Four different data sets formed after data augmentation are summarized in Table 4.6

Table 4.6. Datasets after augmentation

	Orginal Dataset-1[13]	Orginal Dataset-2[14]	Augmented Dataset-1	Augmented Dataset-2
Train-P	1024	180	21.504	3.780
Train-N	1024	180	21.504	3.780
Val-P	100	60	2.100	1.260
Val-N	100	60	2.100	1.260
Test-P	100	60	2.100	1.260
Test-N	100	60	2.100	1.260

Here Train-P shows positive data for the Covid-19 in the training set and Train-N shows negative data. The nomenclature is similar for validation and test data. The experimental results are summarized in Table 4.7. Since the experimental studies were done with input images of different sizes, four different models were tried in the table.

Augmented Dataset-2 is used for training test and validation in Train-37. The results show a

[13] (Soares et al., 2020)
[14] (Yang et al., 2020)

very large training time (more than 3 hours) and a low classification success. The reasons for the long training time are large input size, large filter dimension and big fiter count. Train-38 shows the test results with Augmented Dataset-1 of the network that trained in the previous train. When the results are examined, it is seen that there is no classification success. This is clearly seen in Figure 4.9. Separation of train and validation lines on the graph indicates overfitting.

Train-39 shows the result of the training and testing with Augmented Dataset-1. Here, the input image size is 32x32 and the training time of the network is relatively less compared to image size=64x64. The results show reasonable classification success. Train-40 uses the same network as Train-39, but test data belongs to Augmented Dataset-2. This result indicates that there is no remarkable classification success.

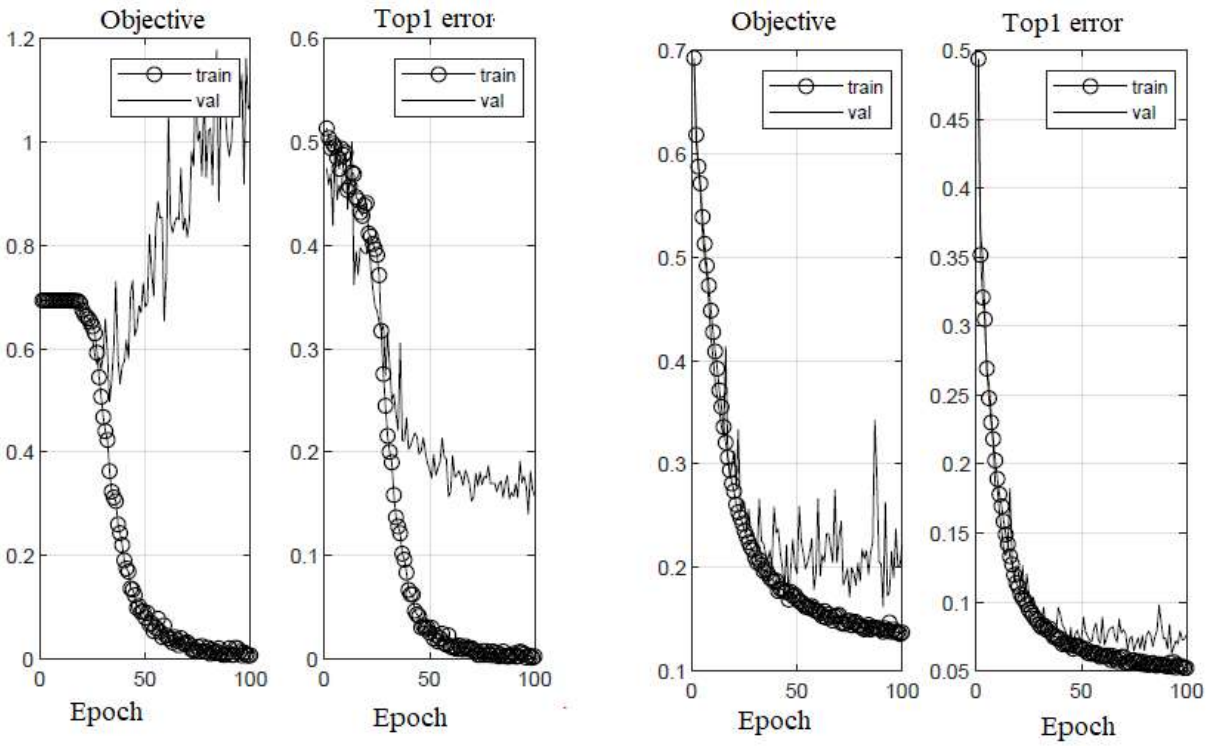

Figure 4.9. First 100 itrations for Train-37 and Train-39

Train-41 uses the same training data and parameters as Train-39. The only difference is that batch size=512. Train-42 shows the test results with the Augmented Dataset-2 of the network trained with Augmented Dataset-1.

Table 4.7. Trainin parameter and results for Train 37-50

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
37	5	64	0,001	100	11.806	0.679	0.679
38[15]	5	64	0,001	100	11.806	0.496	0.476
39	3	32	0.002	100	3.735	0.872	0.871
40[16]	3	32	0.002	100	3.375	0.500	0.667
41	3	512	0.002	100	3.064	0.888	0.890
42[17]	3	512	0.002	100	3.064	0.558	0.591
43	6	64	0.002	100	2.017	0.500	0.666
44	6	64	0.002	300	5.976	0.898	0.929
45	6	64	0.002	500	10.099	0.898	0.930
46[18]	6	64	0.002	500	10.099	0.549	0.612
47	6	128	0.002	100	2.200	0.876	0.8819
48	6	128	0.002	300	4.407	0.933	0.933
49	6	128	0.002	500	10.904	0.934	0.934
50[19]	6	128	0.002	500	10.904	0.588	0.657

When the results highlighted in Table 4.7 are examined, it is seen that there is no acceptable learning for testing with cross dataset. However, it seems that increasing the batch size and new Model-6 provides a small accuracy increase. This could be a clue for future trains. When Train-45 and Train-49 are compared, it is seen that batch size increase does not cause a significant loss of time. It is also seen that increasing the batch size provides a small increase in classification success. This increase is more pronounced between Train-46 and Train-50 where cross-tests were performed. The trainings 47-48-49 was consecutive trainings and was applied as learning rate=0.001 after the 300th iteratition.

4.1.7. Speeding up the Network Training

[15] Trained with Augmented Dataset-2 and tested with Augmented Dataset-1
[16] Trained with Augmented Dataset-1 and tested with Augmented Dataset-2
[17] Trained with Augmented Dataset-1 and tested with Augmented Dataset-2
[18] Trained with Augmented Dataset-1 and tested with Augmented Dataset-2
[19] Trained with Augmented Dataset-1 and tested with Augmented Dataset-2

The training time of the network is an important parameter for deep learning algorithms. Training parameters need to be set appropriately so that network training can be completed in a reasonable time. Some of these parameters can be vital to the success of the network. In this case, the training time of the network can be accelerated by adjusting the parameters that cause relatively less performance degradation. In this section, tips are sought to shorten the network training time.

When Train-37 and Train-45 are compared, it is seen that they have similar training time. During this time, Train-45 completed 500 iterations while Train-37 completed only 100 iterations. Also, Train-37's dataset contains 7560 images, while Train-45's dataset contains 43,008 images. The reason for this situation is the models used in the two trials. The filter sizes in the first layer of Model-5 caused the training parameter number to increase excessively.

Compared to Train-39 and Train-41, batch size=500 for 32x32 images provided a faster training speed and classification success than batch size=100. Therefore, increasing the batch size for small images can increase speed and success.

Training the network with augmented data greatly increases the training time of the network. However, data augmentation is a preferred way to avoid overfitting. In this section, a method is applied to reduce the network training time with augmented data. In this method, the network is first trained with the original data. After sufficient iterations have been provided, the trained network is retrained with augmented data. Training results are presented in Table 4.8. The input image sizes for all the trainings in the table are 100x100.

Network training was performed using Model-7(Table 3.9), Dataset-1 and Augmented Dataset-1. The average iteration time for Dataset-1 is 3 seconds. For Dataset-2, this time is 41.7 seconds. The total training time with Augmented Dataset-1 for 1000 iterations is approximately 41.700 sec. When the first 600 iterations of the training are performed with Dataset-1, the elapsed time is 1,800 seconds. When the remaining 400 iterations are trained with Augmented Dataset-1, the elapsed time is 16,680 seconds. With the applied method, the total training time is 18,480 seconds. If all iterations are done with Augmented Dataset-1, the training time will be 41,700 seconds. The training time difference between the two methods is

approximately 24,900 sec(6 hours and 55 minutes).

Table 4.8. Trainin parameter and results for Train 50-52

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
50	7	512	0,001	0-600	1800	0.805	0.808
51	7	512	0,001	601-1000	16.680	0.902	0.902
52[20]	7	512	0,001	1000	41.700	***	***

When Figure 4.10 is examined, it is seen that the error values increase when the augmented data is passed after the first 600 iterations. However, the error value decreases rapidly in the next iterations.

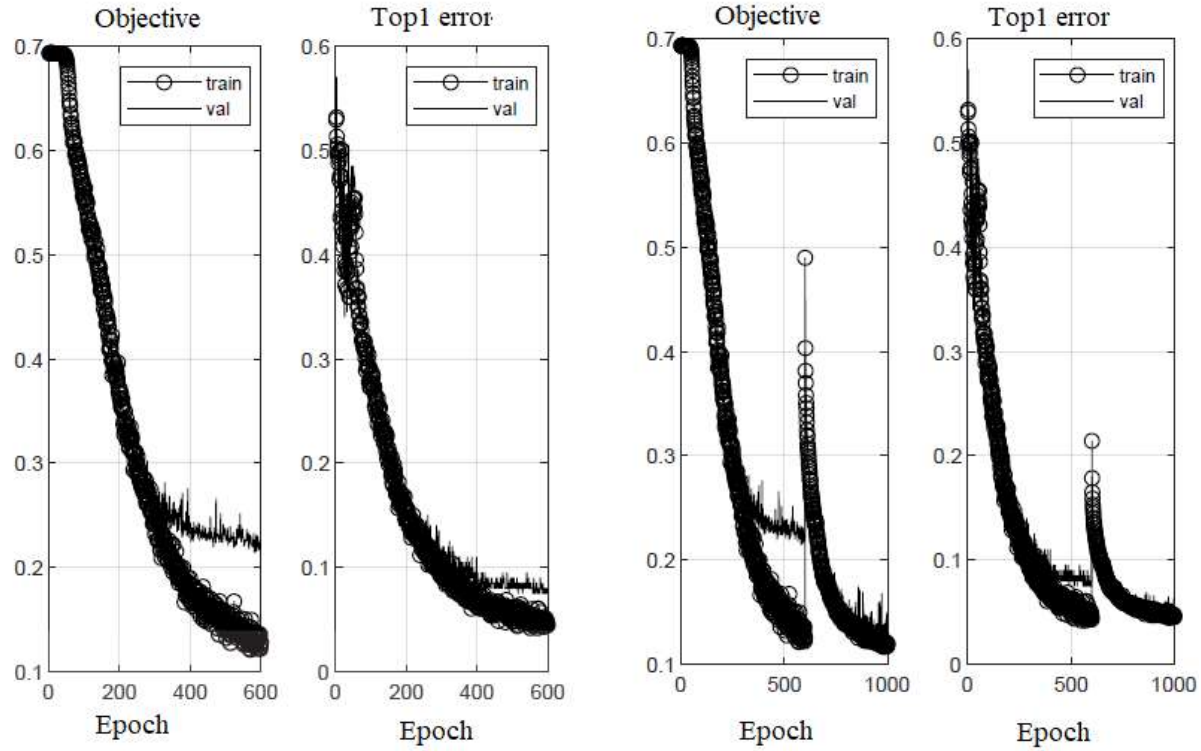

Figure 4.10. First 100 itrations for Train-50 and Train-51

[20] This training was not implemented. It was estimated from the previous two trainings.

The reason for this situation is that the features gained or lost through data augmentation processes change the training parameters. However, since the images do not lose their general qualities, the training parameters of the network quickly adapt to the new situation.

4.1.8. Effects of dataset parsing method

The method of selecting test data has been changed to identify the causes of overfitting in the tests performed up to this point. The first of the new dataset created is which formed by combining the validation and test data in the previously used Original Dataset-1. As a result, the number of test data sets was 200. The training dataset has not been changed. The other dataset is the sequentially selected data that separates the first 200 data in the Original Dataset-1 for the test data.

The aim here is to separate the lung the CT slices of the same individual as training or test data only. Because the chest the CT slices of the same individual are consecutive in the data set. In both cases, batch size=128 and epoch=500. The results are presented in Table 4.9. Since the network and training parameters were the same, the training time was not measured.

Table 4.9 Trainin parameter and results for Train 53-54

Training No	Model No	Batch size	Learning rate	Epoch	Training time(sec)	Accuracy	F1 Scor
53[21]	6	128	0,001	500	-	0.872	0.884
54[22]	6	128	0,001	500	-	0.850	0.841

Training and validation graphs are shown in Figure 4.11. The following conclusions were reached by examining the graph and the table. The success rate is higher and the divergence in the graph is less in the jumped selection method.

[21] Juped selection
[22] Ordered selection

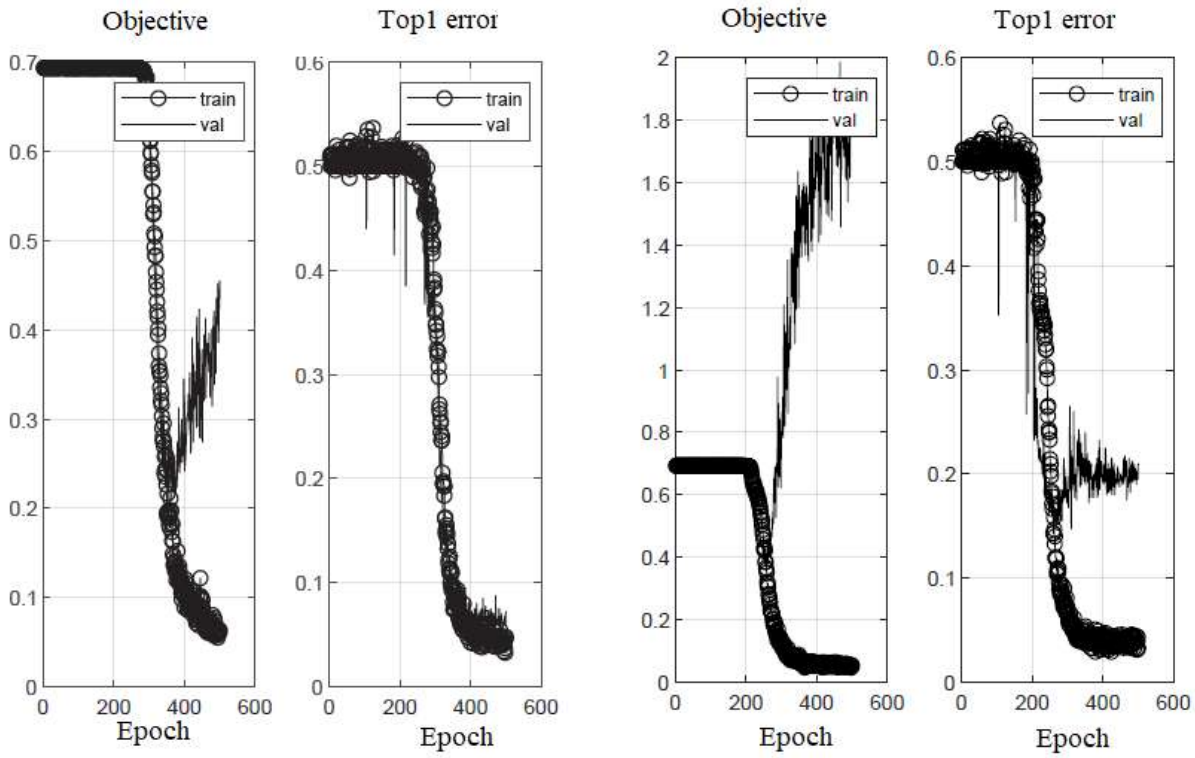

Figure 4.11 Jumped and sequential parsing method

This is because each individual's lung the CT slices are included in the test data. Because these images contain similar global patterns, they caused more overfitting. Global attributes in images are clearly visible in Figure 4.12.

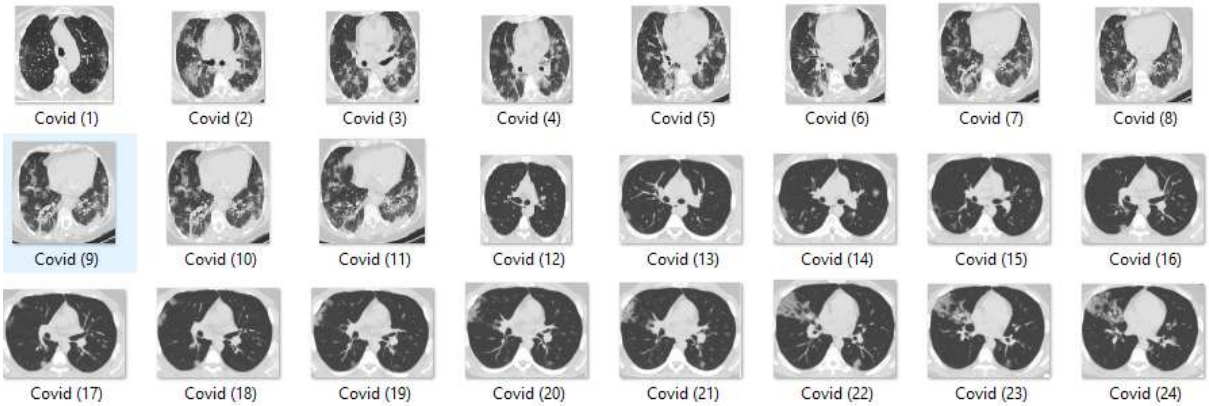

Figure 4.12: Global patterns on the CT image slices

According to this result, separating the CT slices of the same individual as only training or only test data may contribute to obtaining more reliable results.

4.1.9. Creating the most successful model

The biggest problem encountered as a result of experimental studies is overfitting. This is seen in the divergence between the training and validation graphs. In Table 3.10, which was developed as a solution to this problem, input image sizes were increased and data increment was applied 8 times. The transformation process was applied twice, 10 pixels each, in the vertical and horizontal directions. Rotation was applied 5 and 15 degrees in both directions. The same data set was used as validation and test data. The number of training data used for this training is 10161 and the number of test data is 900.

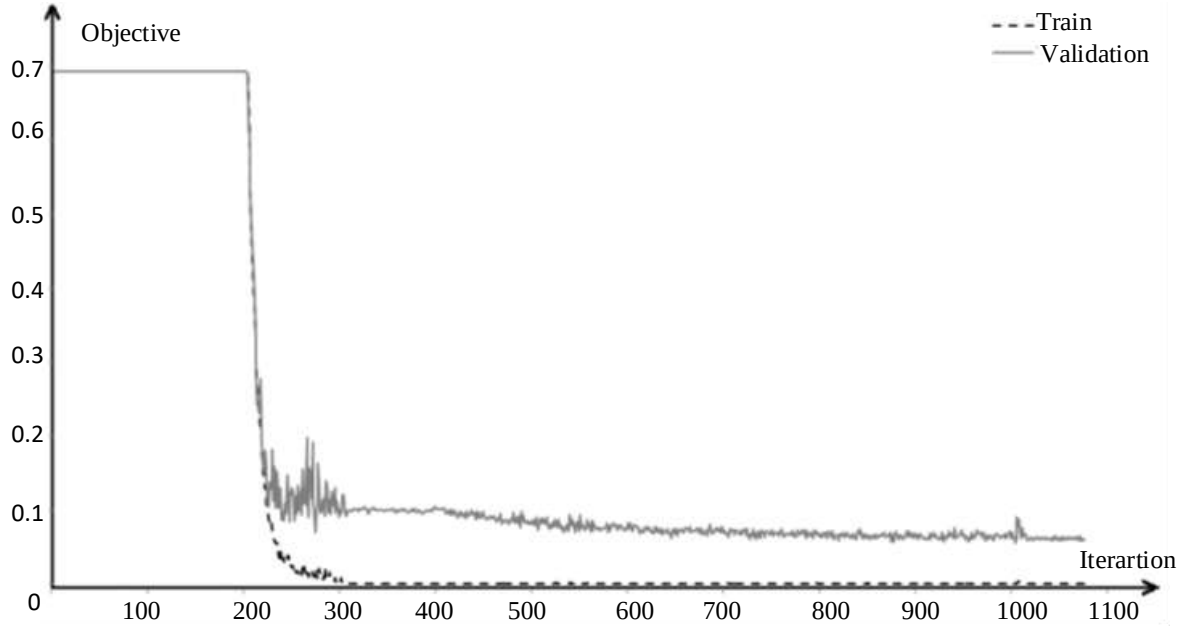

Figure 4.13. Objective functions for Train-55

The training of the network was started with the original dataset. After the 200th iteration, in which the objective function started to decline rapidly, augmented data were included in the training. The learning rate was started with 0.01 and decreased to 0.008. During the training, the batch value was changed between 128-1024. The low learning rate and high batch size were effective in reducing the oscillations seen in the graph. The best accuracy value reached at the 1000th iteration is 0.962 and the F1-score value is 0.963. After this iteration, no increase in classification success was observed.

The following recommendations can be made when this model is compared with the previous ones. Enlarging the input image size is important. Filter sizes should be in order from largest to smallest. Filter numbers can be increased inversely with filter sizes. Thus, the training cost and success of the network can be balanced.

4.2. Comaparative Results

There are many studies in the literature on the diagnosis of the Covid-19 from chest images. These studies vary in terms of data sets, method, classification type and success scales used. A summary of these studies is presented in Table 4.10 to aid in comparison with this thesis work. Studies contain much more data than the data presented in the table. Each study has its own limitations. The data presented here summarizes the common aspects of the related studies with this thesis. The following conclusions can be reached from the examination of the studies.

The value of 0.9738 obtained in the study of Soares, which is the source of the Dataset-1 used in this thesis, is better than the value of 0.963, which is our result.

While the accuracy value obtained in Yang's study, which is the source of Dataset-2 used in this thesis, is 0.89, this value is 0.68 in our thesis, which is very small. The reason for this low result is that the images in the second dataset were compiled from different sources. As a result, the images do not contain common global attributes, so the effect of overfitting is less.

It is seen that studies with high classification success are carried out by adapting pre-trained networks to the diagnosis of the Covid-19(Ismael & Şengür, 2020),(Apostolopoulos & Mpesiana, 2020). These networks are usually multi-layered pre-trained deep CNN networks.

Specially designed deep nets have shown high performance in the diagnosis of the Covid-19.(Marques et al., 2020), (Soares et al., 2020)

Apart from deep CNN networks, classical classification algorithms also give successful results in the diagnosis of the Covid-19.(Yasar & Ceylan, 2021), (Ismael & Şengür, 2020)

Table 4.10 Comparative results

STUDY	DATASET	METHODS-MODELS	RESULTS
(Soares et al., 2020)	1396 lung CT images(386 Covid and 1010 Non covid) are used. Data augmented 5, 10 and 20 times.	23-Layer CNN, k-NN, SVM, Alexnet, MobilnetV2	Results respectively; F1-SCORS: 0.9197, 0.9891, 0.9473, 0.9058, 0.9888 AUC: 0.9404, 0.9901, 0.9599, 0.9284, 0,9903
(Ismael & Şengür, 2020)	380 lung X-ray images(180 Covid and 200 Non covid) are used.	ResNet50+SVM, Fine-tunning of ResNEt50, End-to-end 21-layer CNN model,BSIF+SVM	Accurracy: 0.947, 0.926, 0.916, 0.905
(Apostolopoulos & Mpesiana, 2020)	DS1: 1428 X-ray images(224 Covid, 700 batgerial, 504 normal) DS2: 1428 X-ray images(224 Covid, 700 batgerial, 714 (314 Viral and 400 bacterial), 504 normal)	Different pretrainded CNN models(VGG19, MobileNetv2,Inception, Xception, Inception ResNet v2)	Best Scors: Accurracy: 0.9678 Sensivity: 0.9866 Specificity: 0.9646
(Dansana et al., 2020)	360 X-ray and CT images, (295 Covid, 16 images of SARS, 18 images of streptocus)	VGG-16, Inception_V2, Decision Tree	Result Respectively; Accurracy: 0.91, 0.78, 0.60
(Pham, 2020)	746 CT images (349 Covid from 216 patients and 397 nonCovid)	16 pretrained CNN model used with data augmented and without augmented	Accuracy : 0.962
(Marques et al., 2020)	X-Ray images(500 normal, 504 Pnumonia, 504 Covid-19)	EfficeNetB4 used for binary and multiclass classifications	Accuracy : 0.9949
(Soares et al., 2020)	2482 CT images(1252 Covid and 1230 nonCovid)	eXplainable Deep Learning approach (xDNN)	Accurracy: 0.9738
(Yang et al., 2020)	CT imges from (349 Covid and 463 noCovid)	Multi-task and self-supervised learning	Accuracy : 0.89

5. CONCLUSIONS

Within the scope of this thesis, 55 training applications were made using 8 different CNN models and two different data sets. The data sets are also divided into different subgroups according to the augmentation and seperation methods. Some of these trainings are a continuation of each other, and they are obtained by changing only training parameters such as learning rate and batch size. In others, changes were made to the CNN models. The best results obtained in the applications with the developed models and the data sets in Table 4.6 are as follows.

The created CNN models were first trained and tested with the (Soares et al., 2020) dataset. The best accuracy and F1-score values obtained in the tests are equal to each other and are 0.93. To check the accuracy of these results, the trained network was tested with (Yang et al., 2020) dataset. The best accuracy and F1-scor values obtained according to the cross-test results are 0.59 and 0.66, respectively. Training and test results with Data Set-2 (Yang et al., 2020) yielded 0.68 for accuracy and F1-score values.

A new data set was created as a result of the operations specified in Section 4.1.9 on Soares' data set (Soares et al., 2020). As a result of training with this data set and Model-8 (Table 3.10), the best classification success was 0.96. The findings obtained within the scope of this thesis are as follows.

- The effect of batch size on classification success was not observed.
- ReLU activation function increased the classification success.
- Dropout has made a very small improvement in network success. However, it increased the number of iterations required for training the network.
- The learning rate did not affect the classification success of the network. However, the high learning rate increased the training speed of the network.
- In order to reduce the training time with augmented datasets, it is possible to start the training with the original data and continue with the augmented data after a specified iteration.

- The preferred method of separating the data set consisting of chest the CT slices into training and test data causes overfitting. Here, different slices of the same chest the CT are the source of overfitting as they contain common features.
- For a successful CNN model, filter sizes must be ordered from largest to smallest across the layers.
- In CNN, filter numbers can be increased across layers. With the reduction in image size in the next layers, the features in the image are transferred to the filters. At the same time, the training times across the layers are balanced.
- It is expected that the classification success of the network will increase as the input image resolution increases. However, there was no significant classification success between 32x32 images, 64x64 images and 100x100 images. On the other hand, the best classification success was achieved with 244x244 images.
- In the model that produces the best results, no pooling layer is used.
- With the fuzzy logic application, the classification success of a trained network can be interpreted together with the statistical succes of the network.

6. RECOMMENDATIONS

In this thesis, the Covid-19 pandemic binary diagnosis was made from chest the CT images compiled from open sources with the created CNN model. In the light of the findings of the thesis, the following suggestions can be made to give an idea for the researchers who will conduct individual studies on this subject in the future.

- It takes a lot of time to train deep CNN networks with limited hardware. In such projects, more advanced hardware such as GPU usage and cluster computers should be preferred.
- For higher classification success, familiar pre-trained deep CNN architectures and frameworks should be preferred.
- Instead of presenting the image as a whole to the network and classifying it, it is recommended to detect lesions in the image, segment and classify them probalistic.
- The data set used is decisive in the success of the network. It is recommended that images be homogenized in terms of resolution, contrast, color coding, etc.
- Creating a dataset manually is time consuming. Open source deep learning platforms that can be automatically download and use the data presented in the internet storage environment should be worked with.
- The CNN models created in this thesis were created with the simpleNN method, which is the classical serial CNN application of MatConvNet. For more sophistic models it is recommended to use MatConvNet's dagNN method, which supports parallel layer structures.
- For CNN applications to be developed on a similar subject, it is recommended to reduce the size with filter size, stride and padding parameters instead of using pooling in layers.
- In CNN models, it is recommended to increase the number of filters while decreasing the filter sizes across the layers.
- Because chest the CT slices of the same individual will have the same label, these images should be classified as training data or test data only. Thus, the overfitting effect of the common global features of these slices can be reduced.

- In order to shorten the training time of the network, it is recommended to start the training with the original data and include the augmented data when a specified number of iterations is reached.
- It is recommended to evaluate the success of Model-8, the most successful model developed within the scope of this thesis, by testing it with other datasets and deep learning frameworks.

REFERENCES

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Umar, A. M., Linus, O. U., Arshad, H., Kazaure, A. A., Gana, U., & Kiru, M. U. (2019). Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition. *IEEE Access*, *7*, 158820–158846. https://doi.org/10.1109/ACCESS.2019.2945545

Abraham, A. (2005). Artificial Neural Networks. *Handbook of Measuring System Design, Edited by Peter H. Sydenham and Richard Thorn.* https://doi.org/https://doi.org/10.1002/0471497398.mm421

Ai, T., Yang, Z., Hou, H., Zhan, C., Chen, C., Lv, W., Tao, Q., Sun, Z., & Xia, L. (2020). Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases. *Radiology*, *296*(2), E32–E40. https://doi.org/10.1148/radiol.2020200642

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186

Apostolopoulos, I. D., & Mpesiana, T. A. (2020). *Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks* (pp. 635–640). Physical and Engineering Sciences in Medicine. https://doi.org/https://doi.org/10.1007/s13246-020-00865-4

Ates, O. F., Taydas, O., & Dheir, H. (2020). Thorax Magnetic Resonance Imaging Findings in Patients with Coronavirus Disease (COVID-19). *Academic Radiology*, *27*(10), 1373–1378. https://doi.org/10.1016/j.acra.2020.08.009

Attoh-Okine, N. O. (1999). Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance. *Advances in Engineering Software*, *12*. https://doi.org/10.1016/S0965-9978(98)00071-4

Bahrampour, S., Ramakrishnan, N., Schott, L., & Shah, M. (2015). *Comparative Study of Deep Learning Software Frameworks*. http://arxiv.org/abs/1511.06435

Bai, H. X., Hsieh, B., Xiong, Z., Halsey, K., Choi, J. W., Tran, T. M. L., Pan, I., Shi, L. B., Wang, D. C., Mei, J., Jiang, X. L., Zeng, Q. H., Egglin, T. K., Hu, P. F., Agarwal, S., Xie, F. F., Li, S., Healey, T., Atalay, M. K., & Liao, W. H. (2020). Performance of Radiologists in Differentiating COVID-19 from Non-COVID-19 Viral Pneumonia at Chest CT. *Radiology*, *296*(2), E46–E54. https://doi.org/10.1148/radiol.2020200823

Basheer, I.A., M. H. (200 C.E.). Artificial neural networks: fundamentals, computing, design, and application. In *Journal of Methods Microbiological* (Vol. 11, Issue 3). www.elsevier.com/locate/jmicmeth

Bow, S.-T. (2002). *Pattern Recognition And Image Preprocessing (2Ed , Revised and Expanded)* (p. 720). Marcel Dekker.

Bressem, K. K., Adams, L. C., Erxleben, C., Hamm, B., Niehues, S. M., & Vahldiek, J. L. (2020). Comparing different deep learning architectures for classification of chest radiographs. *Scientific Reports*, *10*(1), 1–16. https://doi.org/10.1038/s41598-020-70479-z

Chung, M., Bernheim, A., Mei, X., Zhang, N., Huang, M., Zeng, X., Cui, J., Xu, W., Yang, Y., Fayad, Z. A., Jacobi, A., Li, K., Li, S., & Shan, H. (2020). CT imaging features of 2019 novel coronavirus (2019-NCoV). *Radiology*, *295*(1), 202–207. https://doi.org/10.1148/radiol.2020200230

Cohen, J. P., Morrison, P., Dao, L., Roth, K., Duong, T. Q., & Ghassemi, M. (2020). *COVID-19 Image Data Collection: Prospective Predictions Are the Future*. 1–38. http://arxiv.org/abs/2006.11988

Dansana, D., Kumar, R., Bhattacharjee, A., Hemanth, D. J., Gupta, D., Khanna, A., & Castillo, O. (2020). Early diagnosis of COVID-19-affected patients based on X-ray and computed tomography images using deep learning algorithm. *Soft Computing, 0123456789*. https://doi.org/10.1007/s00500-020-05275-y

Ding, B., & Qian, H. (2018). Activation Functions and Their Characteristic in Deep Neural Networks. *Collego of Energy and Electrical Engineering*, *6*, 1–6.

Doneva, M. (2020). Mathematical Models for Magnetic Resonance Imaging Reconstruction. *IEEE Signal Processing Magazine, January*, 24–32. https://doi.org/10.1109/MSP.2019.2936964

Hamed, M. A. (2020). An overview on COVID-19: reality and expectation. *Bulletin of the National Research Centre*, *44*(1). https://doi.org/10.1186/s42269-020-00341-9

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. 1–18. http://arxiv.org/abs/1207.0580

Howard, A. G., Wang, W., Zhu, M., Weyand, T., Chen, B., Andretto, M., Kalenichenko, D., & Adam, H. (2012). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*.

Huang, C., Wang, Y., Li, X., Ren, L., Zhao, J., Hu, Y., Zhang, L., Fan, G., Xu, J., Gu, X., Cheng, Z., Yu, T., Xia, J., Wei, Y., Wu, W., Xie, X., Yin, W., Li, H., Liu, M., … Cao, B. (2020). Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. *The Lancet*, *395*(10223), 497–506. https://doi.org/10.1016/S0140-6736(20)30183-5

Ilangovan, S. S., Mahanty, B., & Sen, S. (2016). Biomedical Imaging Techniques. *Information Science Reference (an Imprint of IGI Global), August*, 401–421. https://doi.org/10.4018/978-1-4666-9685-3.ch016

Inoue, H. (2018). Data Augmentation by Pairing Samples for Images Classification. *ArXiv*, *8*.

Inoue, H. (2020). Multi-Sample Dropout for Accelerated Training and Better Generalization. *ArXiv*, *9*.

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., Seekins, J., Mong, D. A., Halabi, S. S., Sandberg, J. K., Jones, R., Larson, D. B., Langlotz, C. P., Patel, B. N., Lungren, M. P., & Ng, A. Y. (2019). CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 590–597. https://doi.org/10.1609/aaai.v33i01.3301590

Işıklı, S. (2007). *Bulanık Mantık ve Bulanık Teknolojiler* (Issue January). https://www.researchgate.net/publication/279657247_Bulanik_Mantik_ve_Bulanik_Teknolojiler

Ismael, A. M., & Şengür, A. (2020). Deep learning approaches for COVID-19 detection based on chest X-ray images. *Expert Systems with Applications*, *29*, 114054. https://doi.org/10.1016/j.eswa.2020.114054

Jeong, H. J., Park, K. S., & Ha, Y. G. (2018). Image Preprocessing for Efficient Training of YOLO Deep Learning Networks. *Proceedings - 2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018*, 635–637. https://doi.org/10.1109/BigComp.2018.00113

Jin, J., Dundar, A., & Culurciello, E. (2015). Flattened convolutional neural networks for feedforward acceleration. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, *2014*, 1–11.

Keskenler, M. F., & Keskenler, E. F. (2017). Geçmişten Günümüze Yapay Sinir Ağları ve Tarihçesi From Past to Present Artificial Neural Networks and History. *Takvim-i Vekayi*, *11*, 8–18.

Khan, S. H., Hayat, M., & Porikli, F. (2018). Regularization of deep neural networks with spectral dropout. *Neural Networks*, *28*. https://doi.org/10.1016/j.neunet.2018.09.009

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*(2). https://doi.org/10.1201/9781420010749

Lecun, Y., Bottou, L., Haffner, P., & Bengio, Y. (1998). Gradient-Based Learning Applied to Document Recognition. *IEEE*, *86*(11), 2279–2324.

Lederer, J. (2021). *Activation Functions in Artificial Neural Networks: A Systematic Overview*. 1–42. http://arxiv.org/abs/2101.09957

Lehtokangas, M., Saarinen, J., Kaski, K., & Huuhtanen, P. (1995). Initializing Weights of a Multilayer Perceptron Network by Using the Orthogonal Least Squares Algorithm. *Neural Computation*, *7*(5), 982–999. https://doi.org/10.1162/neco.1995.7.5.982

Li, Z., Yang, W., Peng, S., & Liu, F. (2020). *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*. http://arxiv.org/abs/2004.02806

Liu, J., Xie, W., Wang, Y., Xiong, Y., Chen, S., Han, J., & Wu, Q. (2020). A comparative overview of COVID-19, MERS and SARS: Review article. *International Journal of Surgery*, *81*(July), 1–8. https://doi.org/10.1016/j.ijsu.2020.07.032

Ma, J., Wang, Y., An, X., Ge, C., Yu, Z., Chen, J., Zhu, Q., Dong, G., He, J., He, Z., Cao, T., Zhu, Y., Nie, Z., & Yang, X. (2021). Toward data-efficient learning: A benchmark for COVID-19 CT lung and infection segmentation. *Medical Physics*, *48*(3), 1197–1210. https://doi.org/10.1002/mp.14676

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., & Wang, Y. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors (Switzerland)*, *17*(4). https://doi.org/10.3390/s17040818

Marques, G., Agarwal, D., & de la Torre Díez, I. (2020). Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. *Applied Soft Computing Journal*, *96*, 106691. https://doi.org/10.1016/j.asoc.2020.106691

Mocanu, I. (2008). An INTRODUCTION TO CUDA Programming. *Journal of Information Systems & Operations Management*, *2*(2), 495–506.

Möllenhoff, K., Oros-Peusquens, A. M., & Shah, N. J. (2012). Introduction to the basics of magnetic resonance imaging. *Neuromethods*, *71*(January), 75–98. https://doi.org/10.1007/7657-2012-56

Moore, C. S., Liney, G. P., Beavis, A. W., & Saunderson, J. R. (2011). A method to produce and validate a digitally reconstructed radiograph-based computer simulation for optimisation of chest radiographs acquired with a computed radiography imaging system. *British Journal of Radiology*, *84*(1006), 890–902. https://doi.org/10.1259/bjr/30125639

Murphy, J. (2016). *An Overview of Convolutional Neural Network Architectures for Deep Learning*. *22*, 1–22. https://www.microway.com/download/whitepaper/An_Overview_of_Convolutional_Neural_Network_Architectures_for_Deep_Learning_fall2016.pdf

Namatēvs, I. (2017). Deep Convolutional Neural Networks: Structure, Feature Extraction and Training. *Information Technology and Management Science*, *20*(1), 40–47.

https://doi.org/10.1515/itms-2017-0007
Narin, A., Kaya, C., & Pamuk, Z. (2020). Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks. *ArXiv Preprint ArXiv:2003.10849*. https://arxiv.org/abs/2003.10849
Nur, A. S., Mohd Radzi, N. H., & Ibrahim, A. O. (2014). Artificial Neural Network Weight Optimization: A Review. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *12*(9). https://doi.org/10.11591/telkomnika.v12i9.6264
O' Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco Hernandez, G., Krpalkova, L., Riordan, D., & Walsh, J. (n.d.). *Deep Learning vs. Traditional Computer Vision*.
O'Shea, K., & Nash, R. (2015). *An Introduction to Convolutional Neural Networks*. http://arxiv.org/abs/1511.08458
Pham, T. D. (2020). A comprehensive study on classification of COVID-19 on computed tomography with pretrained convolutional neural networks. In *Scientific Reports* (Vol. 10, Issue 1). https://doi.org/10.1038/s41598-020-74164-z
Radiuk, P. M. (2018). Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Information Technology and Management Science*, *20*(1), 20–24. https://doi.org/10.1515/itms-2017-0003
Rahman, T., Chowdhury, M. E. H., Khandakar, A., Islam, K. R., Islam, K. F., Mahbub, Z. B., Kadir, M. A., & Kashem, S. (2020). Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray. *Applied Sciences (Switzerland)*, *10*(9). https://doi.org/10.3390/app10093233
Sadowski, P. (2017). Notes on Backpropagation. *Department of Computer Science University of California Irvine*, *1*(2), 1–4. https://www.ics.uci.edu/~pjsadows/notes.pdf
Shi, Y., Wang, G., Cai, X., Deng, J., Zheng, L., Zhu, H., Zheng, M., Yang, B., & Chen, Z. (2020). An overview of COVID-19. *Journal of Zhejiang University-SCIENCE B (Biomedicine & Biotechnology)*, *21*(5), 343–360. https://doi.org/10.1631/jzus.B2000083
Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0
Shuja, J., Alanazi, E., Alasmary, W., & Alashaikh, A. (2021). COVID-19 open source data sets: a comprehensive survey. *Applied Intelligence*, *51*(3), 1296–1325. https://doi.org/10.1007/s10489-020-01862-6
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.
Simpson, S., Kay, F. U., Abbara, S., Bhalla, S., Chung, J. H., Chung, M., Henry, T. S., Kanne, J. P., Kligerman, S., Ko, J. P., & Litt, H. (2020). Radiological Society of North America Expert Consensus Statement on Reporting Chest CT Findings Related to COVID-19. Endorsed by the Society of Thoracic Radiology, the American College of Radiology, and RSNA - Secondary Publication. *Journal of Thoracic Imaging*, *35*(4), 219–227. https://doi.org/10.1097/RTI.0000000000000524
Singh, D., Kumar, V., Vaishali, & Kaur, M. (2020). Classification of COVID-19 patients from chest CT images using multi-objective differential evolution–based convolutional neural networks. *European Journal of Clinical Microbiology and Infectious Diseases*, *39*(7), 1379–1389. https://doi.org/10.1007/s10096-020-03901-z
Singnoo, J., & Finlayson, G. D. (2010). Understanding the gamma adjustment of images. *Final Program and Proceedings - IS and T/SID Color Imaging Conference*, *June*, 134–139.
Sivanandam, S. N., Sumathi, S., & Deepa, S. N. (2007). Introduction to fuzzy logic using

MATLAB. In *Introduction to Fuzzy Logic using MATLAB*. https://doi.org/10.1007/978-3-540-35781-0

Soares, E., Angelov, P., Biaso, S., Froes, M. H., & Abe, D. K. (2020). SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification. *MedRxiv*, 1–8.

Sudeep, K. S., & Pal, K. K. (2016). Preprocessing for image classification by convolutional neural networks. *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2016 - Proceedings*, 1778–1781. https://doi.org/10.1109/RTEICT.2016.7808140

Sun, W., Brown, S. B., & Leach, R. K. (2012). An overview of industrial X-ray computed tomography. *National Phsical Laboratory, January*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions Christian. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–5. https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

Tilborghs, S., Dirks, I., Fidon, L., Willems, S., Eelbode, T., Bertels, J., Ilsen, B., Brys, A., Dubbeldam, A., Buls, N., Gonidakis, P., Sánchez, S. A., Snoeckx, A., Parizel, P. M., de Mey, J., Vandermeulen, D., Vercauteren, T., Robben, D., Smeets, D., … Suetens, P. (2020). Comparative study of deep learning methods for the automatic segmentation of lung, lesion and lesion type in CT scans of COVID-19 patients. *ArXiv*.

Tosunoğlu, İ. C. (2009). Glokom Hastalığı Tanı ve İzlemede Grüntü İşleme ve Bilgi Teknolojileri. In *Kocaeli University-Institute of Science and Technology* (Vol. 5, Issue June). Kocaeli University.

Vedaldi, A., Lenc, K., & Gupta, A. (2015). MatConvNet: Convolutional neural networks for MATLAB. *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, 689–692. https://doi.org/10.1145/2733373.2807412

Vinh, D. B., Zhao, X., Kiong, K. L., Guo, T., Jozaghi, Y., Yao, C., Kelley, J. M., & Hanna, E. Y. (2020). Overview of COVID-19 testing and implications for otolaryngologists. *Head and Neck*, *42*(7), 1629–1633. https://doi.org/10.1002/hed.26213

Wang, J., Ma, Y., Zhang, L., Gao, R. X., & Wu, D. (2018). Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, *48*, 144–156. https://doi.org/10.1016/j.jmsy.2018.01.003

Wu, J. (2017). Introduction to Convolutional Neural Networks. In *Introduction to Convolutional Neural Networks* (pp. 1–31). National Key Lab for Novel Software Technology-Nanjing University. https://cs.nju.edu.cn/wujx/paper/CNN.pdf

Yam, J. Y. F., & Chow, T. W. S. (2000). A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing*, *30*(1–4), 219–232. https://doi.org/10.1016/S0925-2312(99)00127-7

Yamashita, R., Nishio, M., Togashi, K., & Do, richard K. G. (2018). Convolutional neural networks: an overview and application in radiology. *Springer*, *19*, 21–30. https://doi.org/10.1007/s13244-018-0639-9

Yan, Q., Wang, B., Gong, D., Luo, C., Zhao, W., Shen, J., Shi, Q., Jin, S., Zhang, L., & You, Z. (2020). COVID-19 Chest CT image segmentation – A deep convolutional neural network solution. *ArXiv*, 1–10.

Yang, X., He, X., Zhao, J., Zhang, Y., Zhang, S., & Xie, P. (2020). *COVID-CT-Dataset: A CT Scan Dataset about COVID-19*. 1–14. http://arxiv.org/abs/2003.13865

Yasar, H., & Ceylan, M. (2021). A novel comparative study for detection of Covid-19 on CT

lung images using texture analysis, machine learning, and deep learning methods. *Multimedia Tools and Applications*, *80*(4), 5423–5447. https://doi.org/10.1007/s11042-020-09894-3

Zhang, Y.-D., Satapathy, S. C., Zhu, L.-Y., Gorriz, J. M., & Wang, S.-H. (2020). A seven-layer convolutional neural network for chest CT based COVID-19 diagnosis using stochastic pooling. *IEEE Sensors Journal*, *XX*(Xx), 1–1. https://doi.org/10.1109/jsen.2020.3025855