# DESIGN OF AN OFFLINE OTTOMAN CHARACTER RECOGNITION SYSTEM FOR TRANSLATING PRINTED DOCUMENTS TO MODERN TURKISH

**A Thesis Submitted to**
**the Graduate School of Engineering and Sciences of**
**İzmir Institute of Technology**
**in Partial Fulfillment of the Requirements for the Degree of**

**MASTER OF SCIENCE**

**in Electronics and Communication Engineering**

**by**
**Naz KÜÇÜKŞAHİN**

**December 2019**
**İZMİR**

# ACKNOWLEDGMENTS

First and above all, I would like to express my sincere gratitude to my thesis supervisor Assist. Prof. Dr. Mehmet Zübeyir ÜNLÜ for giving me the chance to work under his guidance. His enthusiasm and immense knowledge have deeply inspired me.

Secondly, I owe my deepest gratitude to my beloved parents, Şerife Nur ÖZBATIR and Recep Semih ÖZBATIR. They have always been my biggest source of motivation.

I am deeply grateful to my dear husband, Oğuzhan KÜÇÜKŞAHİN who kept me going with his encouragement and never-ending patience. I could never have completed my thesis studies without his continuous support.

Finally, I must thank to my friends and my colleagues who motivate me throughout my thesis studies.

# ABSTRACT


## DESIGN OF AN OFFLINE OTTOMAN CHARACTER RECOGNITION SYSTEM FOR TRANSLATING PRINTED DOCUMENTS TO MODERN TURKISH

Optical character recognition (OCR) is one of the most studied topics for many years. As a result of these studies, systems developed especially for the Latin alphabet have become more accurate even for handwritten texts. However, there are very limited studies on Ottoman OCR systems in the literature and it is still a subject of interest due to the complexity of the language in grammar, writing and spelling.

In this thesis, it is aimed to design an offline OCR system that recognizes Ottoman characters using deep convolutional neural networks. The proposed work consists of several steps such as image processing, image digitization and character segmentation, adaptation of inputs to the network, training of the network, recognition and evaluation of results.

Firstly, a character dataset was created by segmenting text images of different lengths that was selected among scanned samples of various Ottoman literature from the digital database of Turkish National Library. Two convolutional neural networks of different complexity were trained with the created character dataset and the relationship between recognition rates and network complexity was evaluated.

Secondly, using the Histogram of Oriented Gradients and Principal Component Analysis, the features of the created dataset were extracted and the Ottoman characters were classified with k-Nearest Neighbor Algorithm and Support Vector Machines which are widely used classification methods in the literature.

The performed analyzes have shown that both networks provide acceptable recognition rates compared to the conventional classifiers, however complex deep neural network showed better accuracy and lower loss.

# ÖZET

## BASILI DÖKÜMANLARIN MODERN TÜRKÇEYE ÇEVRİLMESİ İÇİN ÇEVRİMDIŞI OSMANLICA KARAKTER TANIMA SİSTEMİ TASARIMI

Optik karakter tanıma yıllardır üzerinde en çok çalışma yapılan konulardan bir tanesidir. Bu çalışmaların sonucunda, özellikle Latin alfabesi için geliştirilen sistemler el yazısı metinler için bile iyi tanıma sonuçları gösterir hale gelmişlerdir. Ancak literatürdeki Osmanlıca optik karakter tanıma sistemleri için yapılan çalışmalar oldukça sınırlıdır. Gramer, yazma ve heceleme gibi konulardaki karmaşıklığından dolayı Osmanlıca, optik karakter tanıma alanında ilgi çekiciliğini hala korumaktadır.

Bu çalışmada, derin evrişimsel sinir ağları kullanılarak Osmanlıca için çevrimdışı karakter tanıma sistemi tasarlanması amaçlanmıştır. Yapılan çalışma, görüntü işleme, taratılan metinlerin sayısallaştırılarak karakter parçalarına ayrılması, Osmanlıca karakter veri setinin oluşturularak sinir ağına uyarlanması, sinir ağının eğitilmesi, karakterlerin tanınması ve karakter tanıma oranlarının değerlendirilmesi adımlarından oluşmaktadır.

Bu amaçla ilk olarak, Milli Kütüphane'nin sayısal veritabanından seçilmiş çeşitli Osmanlıca eserlerden alınan farklı uzunluktaki taranmış metin görüntüleri bölütlenerek karakter veri seti oluşturulmuştur. Farklı karmaşıklıktaki iki evrişimsel sinir ağı, oluşturulan karakter seti ile eğitilmiş ve tanıma oranları ile ağ karmaşıklığı ilişkisi değerlendirilmiştir.

Daha sonra Yönelimli Gradyanların Histogramı ve Temel Bileşen Analizi kullanılarak oluşturulan veri setinin öznitelikleri çıkarılmış ve Osmanlıca karakterler bu öznitelikler kullanılarak literatürde yaygın olarak uygulanan sınıflandırma yöntemlerinden, k-En Yakın Komşu Algoritması ve Destek Vektör Makineleri ile sınıflandırılmıştır.

Yapılan performans analizleri, her iki ağın da geleneksel sınıflandırıcılara kıyasla daha iyi tanıma oranlarına sahip olduğunu göstermiş, bununla beraber karmaşık derin evrişimsel sinir ağının en yüksek tanıma oranına sahip olduğunu ortaya koymuştur.

# TABLE OF CONTENTS

# LIST OF FIGURES

**Figure** **Page**

**Figure** **Page**

**Figure** **Page**

# LIST OF TABLES

**Table** **Page**

# CHAPTER 1

# INTRODUCTION TO OPTICAL CHARACTER RECOGNITION

Optical character recognition (OCR) is the field of identifying characters that belong to certain alphabet from its source media. Image of a document is the main source of OCR systems and it may contain machine printed text, handwritten text or even text written in cursive script. An OCR system aims to extract characters and convert them into a form such that computers can understand text information within any document image.

As shown in Figure 1, there are two types of OCR systems in general, offline systems and online systems.

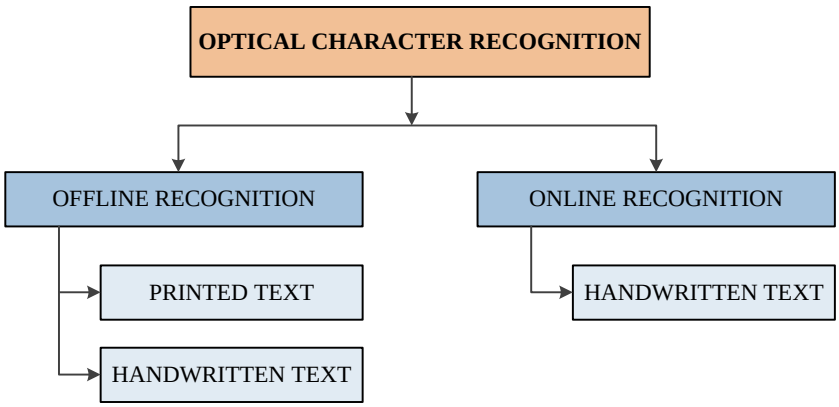



Figure 1. Types of Optical Character Recognition

In online OCR systems, recognition is achieved in real time while character or text is being written through an input device such as touch screen or gesture sensing pens. However, in offline OCR systems, document image must be uploaded to a computer and after the necessary operations are applied on the text, results are obtained after certain amount of processing time.

OCR systems consist of multiple stages such as image acquisition, pre-processing, segmentation and application of feature extraction and classification methods.

Font type, quality of a scanning device, and paper type are some of the factors that influence the performance of an OCR system. In order to eliminate effects of such factors and achieve better system performance, various image-processing operations are applied in pre-processing stage. Segmentation process refers to extracting text in the document to the smaller text pieces in the order of lines, words, and then to characters. After segmentation stage is completed, feature extraction stage is conducted for finding discriminative features that minimize variability within and maximize variability between classes.

By using extracted features, segmented characters are mapped into different classes in the classification stage (Islam, Islam, and Noor, 2016).

Many real life examples can be mentioned on OCR field such as separating letters according to the postal codes written on them, recognition of banking checks for automated account transactions, transferring libraries to digital databases, and recognition of license plates for security purposes etc.

OCR is a well-studied area and there are excessive number of researches conducted to date especially in Latin, Chinese and Arabic alphabets. For example, (Sarfraz, Nawaz, and Al-Khuraidly, 2003) presented an offline Arabic text recognition system which preprocesses text first by applying noise removal and drift correction, then segments it into lines according to baselines and into characters using vertical projection profiles. Moment invariant technique was used for feature extraction and about 73% recognition rate was obtained by using radial basis function (RBF) network.

However, due to the complexity of the language in grammar, writing and spelling, it is noteworthy that there were not enough OCR studies on the Ottoman alphabet, and that the studies did not reach a satisfactory level of accuracy especially in handwritten character recognition.

## 1.1. OCR Studies on the Ottoman Alphabet in the Literature

Noticeable studies on the Ottoman character recognition have been started in the late 90s and continued to date.

In the study of (Öztürk, Güneş, and Özbay, 2000), machine-printed Ottoman characters were used to train the multilayer feedforward network with backpropagation and classification accuracy of 95% was reported.

Approaches based on content search in Ottoman documents mainly focuses on searching a small image in an entire document image. (Şaykol et al., 2004) and (Ataer and Duygulu, 2006) studied such systems and they aimed to identify segmented character groups within a document image instead of identifying individual characters by using matching technique based on word length similarity and vertical projection profiles.

(Onat, Yıldız, and Gündüz, 2008) presented an OCR system for handwritten Ottoman scripts that applies left-right Hidden Markov Models (HMM) for identification. Proposed system showed 65% accuracy.

In the study conducted by (Kurt, Turkmen, and Karsligil, 2007), linear discriminant analysis (LDA) was applied in order to emphasize distinctive features while reducing dimensionality of the images. In classification step, test character was considered to belong to the nearest class and distances between them were determined by Euclidian Distance.

(Kılıç et al., 2008) aimed to recognize machine printed Ottoman scripts by using Support Vector Machines and trained it using linear, quadratic and Gaussian RBF kernels. Study showed that quadratic kernel gives the best recognition rate with 87.32%.

(Yalnız et al., 2009) also represented content-based retrieval approach for digital Ottoman archives. The proposed system allows character recognition directly from digitized images and supports content query both by pieces of word images or by words entered by user through a virtual keyboard. The system uses sliding-window and histogram methods for segmentation and combines them with several recognition approaches such as neural networks and graph-based model.

An online handwritten ottoman character recognition system was proposed by (Nalbant, Burunkaya, and Eroğlu, 2009) and they aimed to recognize characters drawn by a mouse. A unique direction code was assigned to each movement in the coordinate plane, and a code was derived from the movement of the mouse during writing. If there is a match between original character codes and newly created character, it is said to be recognized by the system.

(Can et al., 2010) proposed similar approach for an offline system however they focused on Ottoman poetry in particular in order to detect repetitive words at the end of each line which is known as “redif” in Ottoman literature.

(Arifoğlu and Duygulu, 2011) calculated the word profiles, pixel transitions from background to text and vertical projection of each word and used Dynamic Time Warping in order to find these feature vector's distance to each other. In addition, the study showed that shape context descriptor could also be used for word matching.

Due to its interconnected characteristics, segmentation of Ottoman text is difficult compared to others. (Adıgüzel, Şahin, and Duygulu, 2012) presented an approach that combines connected component based and projection based methods for line segmentation. Further, Fourier Curve Fitting is applied on projections, in order to prevent incorrect separation of lines.

(Şahin et al., 2012) proposed an end user interface for Ottoman documents rather than an OCR system. With the help of this interface, new Ottoman documents can be saved to the database, binarized and segmented into words. Also, through the same interface, labeling processes that require expert knowledge can be realized, and word search on tagged texts and image search on untagged texts can be performed.

The studies mentioned above are based on traditional machine learning approaches. Although studies about deep learning have been continued for a while, easy access to massive sets of labeled data and computing power achieved today makes it possible to perform deep learning applications and it is started to be applied on optical character recognition area also. Deep learning showed promising performance in optical character recognition, handwriting recognition, object recognition, speech recognition, image recognition, text mining etc.

## 1.2. Deep Learning-Based Ottoman and Arabic Character Recognition Approaches in the Literature

Although there are not enough studies about the application of deep learning in Ottoman Turkish alphabet, there are studies in Arabic, which is similar to Ottoman alphabet. Some of the prominent deep learning-based Arabic and Ottoman character recognition studies are as follows.

(Aydemir et al., 2014) compared HMM and recurrent neural networks (RNN) by applying both method to Turkish and Ottoman Turkish. Study showed that RNN method showed 8% higher accuracy in both datasets.

(Ashiquzzaman and Tushar, 2017) also studied convolutional neural networks (CNN) but they focused on Arabic numeral recognition. The proposed model showed 97.4% percent accuracy.

(Younis and Khateeb, 2017) presented a deep CNN for the handwritten Arabic character recognition and applied batch gradient descent method for optimizing weights. Proposed system was trained and tested with two different datasets (AIA9k, AHCD). Accuracies of 94.8% and 97.6% were obtained respectively. In the study of (Elsawy, El-Bakry, and Loey, 2017), a deep CNN was built from multiple convolutional layers that were followed by linear rectifiers and pooling layers. 16800 of handwritten Arabic characters were used to train and test the proposed network. 5.1% misclassification error was reported in the study.

(Ahmed et al., 2017) studied convolutional neural networks in order to recognize Arabic texts from natural scene images. 3x3 and 5x5 sized feature detectors were used in the convolutional layer and network was trained with distinct learning rates.

(Ali, Pickering, and Shafi, 2018) also presented an OCR system based on CNNs for recognizing isolated Urdu characters from natural scene images. They manually created a dataset by cropping 14000 isolated Urdu characters and split them by 70% to 30% for training and testing. Study showed that best error rate, 11.32%, was achieved when they mix filter sizes, use 2x2 pooling layer after and choose learning rate to 0.005.

## 1.3. The Aim of the Thesis

A great archive that contains billions of pages has been inherited from the Ottoman Empire to the present day with great care, which involves history and culture of 23 present-day countries in Europe, North Africa and the Middle East.

Turkey has several institutions such as Turkish National Library, Istanbul Metropolitan Municipality Atatürk Library, Marmara University, Turkish State Archives, and Research Center for Islamic History, Art and Culture etc. These institutions host millions of literary work, official state documents, and newspapers written in Ottoman Turkish. Most of these documents have already digitized with great care and are being presented publicly. However, even if we assume that all currently available Ottoman documents are somehow digitized, there is no way for a researcher or a student to be able

to access and search whatever he or she needs to find in some Ottoman literature because the documents are digitized and kept in image format.

This study aims to propose an OCR method that transfers documents written in Ottoman Turkish to the electronic world. In simple terms, the proposed method can be used in a system that takes a document image as an input, and then converts it to a text searchable word document in contemporary Turkish. Thus, institutions mentioned above can transfer their Ottoman archive easily with this system and present a search database for those who demand. The proposed system can be developed even further and serve as a Google-like online search database for Ottoman contents. By this means, a language that is no longer in use today but has great historic importance in our country will be transferred to next generations.

## 1.4. The Organization of the Thesis

This thesis study is organized as follows; Chapter 2 introduces the Ottoman Alphabet. The scientific and technical information about Deep Learning and Convolutional Neural Networks, and applied approaches are presented in Chapter 3. Experimental work and their results of all implemented methods in the study, and performance analyses of the methods applied are given in Chapter 4. Finally, discussions and conclusions are presented in Chapter 5.

# CHAPTER 2

# OTTOMAN ALPHABET

The Ottoman Empire had been covered an area of about 5.6 million square km from 1300 to 1922. Ottoman Turkish is a written language that includes Arabic and Persian elements and it is widely used in Anatolia and the places where the Ottoman Empire reigned.

Even though Ottoman Turkish had not been actively used since the end of the Empire, a great number of written document such as public records, state archives, land certificates, gravestones and most importantly literatures were transferred to present day. Therefore, Ottoman Turkish maintains its importance even today.

Along with the 28 letters in Arabic alphabet, the letters پ, چ and ژ from Persian alphabet had also been added to the Ottoman alphabet. With other additions, there are 35 letters in total and while 34 of them can be at the end of a word, 25 of them can be in the middle and only 25 of them can be at the beginning of a word. Other than letters, digits, punctuation marks, spaces and special symbols are used. Some letters may have exactly the same shape but they can be separated from one to another by the addition of complementary characters like position and number of points integrated with the letter.

In Ottoman alphabet, letters vary with their body shape, number of dots they have and position of these dots relative to main body. Dots can be located below the main shape or they are located above. The dots may be in groups of one, two or three. Every dotted letter also has a non-dotted one.

Similar to Arabic alphabet, Ottoman Turkish is written from right to the left. Letters change shape regarding whether if they are at the beginning, in the middle or at the end of a word. As opposed to letters, numbers are written from left to right and there are no strict rules for punctuation marks. Numbers are shown in Table 1 and letter types are shown in Table 2.

As can be seen from Table 2, vowels in the Ottoman alphabet is inadequate when compared to the modern Turkish alphabet. For example, both “a” and “e” vowels in

Turkish alphabet covered by letter ا in Ottoman alphabet and letter و covers for “u”, ”ü”, “o” and “ö” vowels.

Table 1. Representations of Numbers in Ottoman

| **Arabic Form** | ٠ | ١ | ٢ | ٣ | ٤ | ٥ | ٦ | ٧ | ٨ | ٩ |
|---|---|---|---|---|---|---|---|---|---|---|
| **Number** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Table 2. The Ottoman Alphabet and Its Equivalent in Latin
(Source: Ottoman Turkish, 2010)

| **Initial** | **Medial** | **Final** | **Isolated** | **Latin Form** |
|---|---|---|---|---|
| ا | ـا | ـا | ا | omitted |
| بـ | ـبـ | ـب | ب | b |
| پـ | ـپـ | ـپ | پ | p |
| تـ | ـتـ | ـت | ت | t |
| ثـ | ـثـ | ـث | ث | s̲ |
| جـ | ـجـ | ـج | ج | c |
| چـ | ـچـ | ـچ | چ | ç |
| حـ | ـحـ | ـح | ح | ḥ |
| خـ | ـخـ | ـخ | خ | ḫ |
| د | ـد | ـد | د | d |

(cont. on next page)

Table 2. (cont.)

| ذ | ذ | ذ | ذ | z̲ |
|---|---|---|---|---|
| ر | ر | ر | ر | r |
| ز | ز | ز | ز | z |
| ژ | ژ | ژ | ژ | j |
| سـ | سـ | س | س | s |
| شـ | شـ | ش | ش | ṣ |
| صـ | صـ | ص | ص | ṣ |
| ضـ | ضـ | ض | ض | ż |
| طـ | طـ | ط | ط | ṭ |
| ظـ | ظـ | ظ | ظ | ẓ |
| عـ | عـ | ع | ع | ‘ (ayn) |
| غـ | غـ | غ | غ | ġ |
| فـ | فـ | ف | ف | f |
| قـ | قـ | ق | ق | ḳ |
| كـ | كـ | ك | ك | k |
| گـ | گـ | گ | گ | g |
| ـ | ڭـ | ڭ | ڭ | ñ |
| لـ | لـ | ل | ل | l |
| مـ | مـ | م | م | m |

(cont. on next page)

Table 2. (cont.)

| | | | | |
|---|---|---|---|---|
| نـ | نـ | ن | ن | n |
| و | و | و | و | v |
| هـ | هـ | ـة ه | ة ه | h |
| يـ | يـ | ى | ى | y |

Ottoman Turkish is written cursively and letters of the Ottoman alphabet can be divided into two divisions as connected and unconnected letters. The connected letters may be joined to the letters that follow them. The unconnected letters do not join with the letter after. Such isolated letters are وارزژد. When they occur the word is broken; that is, the pen is taken up and the second part of the word is resumed unconnected. Thus, they only appear at the end of the first sub word and next letter forms as the initial letter of the second sub word (Harfler, 2019).

In addition, there are many writing styles in the Ottoman Turkish and Ruq'a, Naskh, Thuluth and Taliq are the most common ones.

Ruq'a is the ordinary handwriting used in letters and in all kinds of civil and official documents. It has been developed as a convenient writing for daily correspondence.

Figure 2. Example of Thuluth Writing Style
(Source: Tulum, 2009)

Naskh is the common print and used mostly for religious books. It also includes the auxiliary signs called Hareke. It was preferred when accurate and easy reading were aimed.

Thuluth is the larger version of Naskh. It is an ornamental script mostly used in mosques, tombs, inscriptions and plates, printed book covers, newspaper headlines etc. Figure 2 shows an example of Thuluth writing style.

Taliq is the Persian model of Arabic characters. General place of use of Taliq was the documents of the Ottoman courts and it was also used on stone inscriptions for art purposes (Tulum, 2009).

# CHAPTER 3

# DEEP LEARNING AND CONVOLUTIONAL NEURAL NETWORKS

## 3.1. Introduction to Deep Learning

Artificial intelligence is the ability of a computer to behave like a human brain. It can be defined as a set of predetermined rules for a machine to complete a certain task. Machine learning is a sub-branch of artificial intelligence and the main purpose of machine learning algorithms is to make accurate predictions by learning on their own according to the information coming to the system. The efficiency and accuracy of a machine-learning algorithm depends on how well it is trained (Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences, 2018).

Training computers to function like human brain is usually performed by using artificial neural networks. They are modeled with the inspiration from human nervous system.

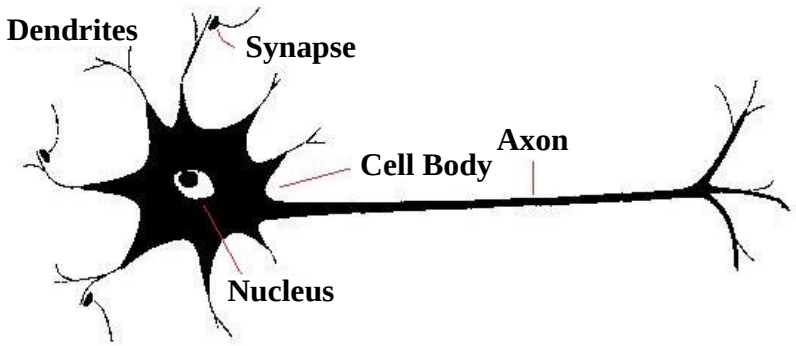



Figure 3. A Biological Neuron
(Source: Jain et al., 1996)

As shown in Figure 3, a biological neuron consists of a nucleus, branches that are called dendrites, and a tail called axon. While dendrites receive signals into the neuron, axon transmits them.

Moreover, synapse is the part that where the whole concept of signal transmission from neuron to neuron occurs. The human brain consists of countless neurons and it continually tries to process, interpret and categorize the new information it receives by comparing it to information that has already known (Jain, Mao, and Mohiuddin, 1996).

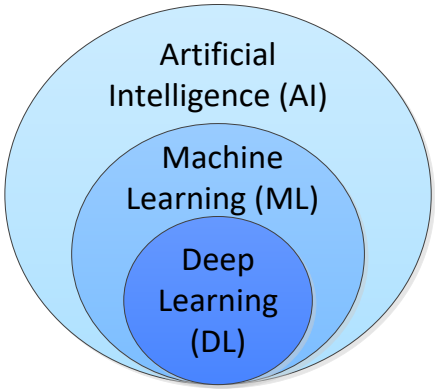



Figure 4. Relation between AI, Machine Learning and Deep Learning (Source: Clearing the Confusion: AI vs. Machine Learning vs. Deep Learning Differences, 2018)

Deep learning however, is a sub-branch of machine learning and it is sometimes referred to as deep neural networks. Figure 4 shows the relationship between artificial intelligence, machine learning and deep learning.

In deep learning, there is no need for predefined rules, any guidance or feature extraction in order to classify new information. Unlike machine learning, this process is done automatically when network is exposed to large amounts of data.

Furthermore, in order to process large datasets, computers with higher processing power are needed in deep learning compared to machine learning (Clearing the Confusion: AI vs. Machine Learning vs Deep Learning Differences, 2018).

## 3.2. Concept of Artificial Neuron

(Mcculloch and Pitts, 1990) introduced the concept of artificial neuron. In its very basic form, a neural network only has a single binary threshold unit that is composed of an input and an output layer.

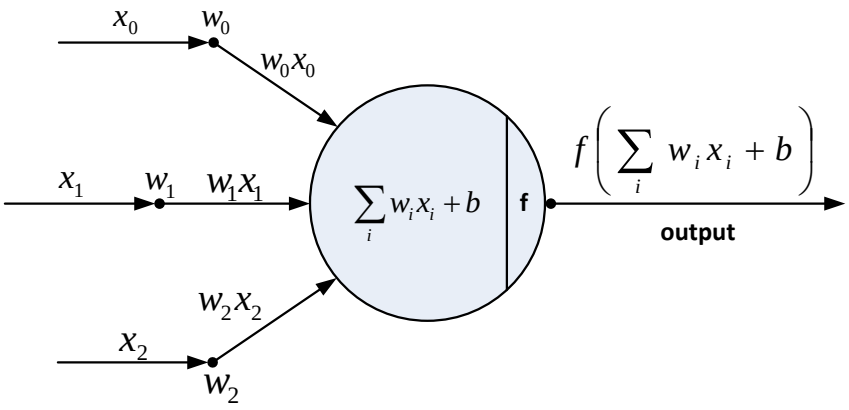



Figure 5. Artifical Neuron
(Source: Karpathy, 2016)

Artificial neuron shown in Figure 5 has three inputs. The weighted sums of these inputs plus the bias are fed to the unit, and then the output layer is calculated. The output can be zero or one regarding to the weighted sum is less or greater than some threshold value. The basic mathematical model can be written as:

$$output = \begin{cases} 0 \text{ if } \sum_i w_i x_i \leq threshold \\ 1 \text{ if } \sum_i w_i x_i > threshold \end{cases}$$

By using the bias and writing weighted sum as a dot product, artificial neuron model can be rewritten as:

$$output = \begin{cases} 0 \text{ if } \boldsymbol{w} \cdot \boldsymbol{x} + b \leq 0 \\ 1 \text{ if } \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \end{cases}$$

According to the weights, the network decides what information is important, what is not, what information will be passed along, and what will not. The weights are assigned randomly at the beginning. When the network is being trained, they are adjusted by measuring how far the output is from the expected outcome.

Most of the time, even though it has many inputs, one neuron may not be sufficient. Neurons operating in parallel are called as layer.

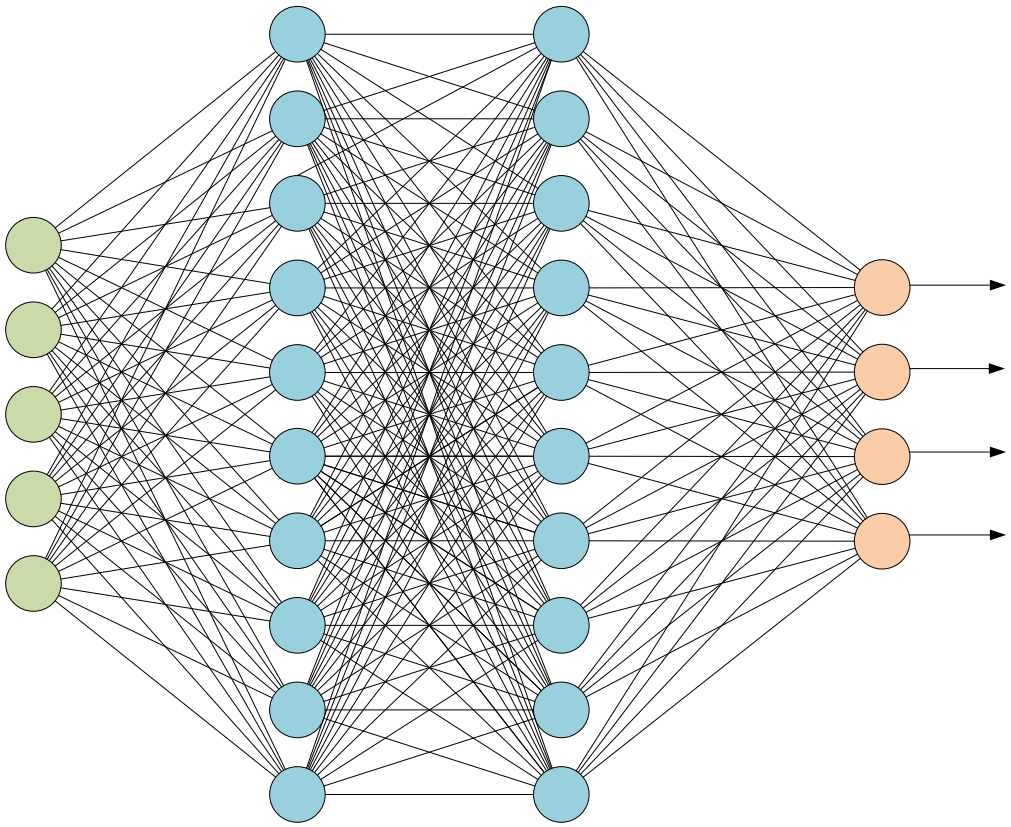


Figure 6. Layers of Deep Neural Network
(Source:Training Deep Neural Networks, 2018)

As can be seen in Figure 6, data is modelled by successive layers in a deep neural network, and the number of these layers specifies the network's "depth". In a deep neural network, output layer can be binary, continuous or categorical variable depending on the application.

## 3.3. Activation Functions

In order for the network to measure its performance on the training data and to steer the weights towards correct direction, an activation function is applied to the weighted sum of the inputs. Activation function performs non-linear transformations over the weighted sum and without it; a neural network will just behave as linear regression with limited learning capability. Depending on the application, different types of activation functions are used.

### 3.3.1. Threshold Function

*x*-axis is the weighted sum of inputs and the *y*-axis has values from 0 to 1. As shown in Figure 7, threshold function will give the output 0 if the input value is less than 0. If it is more than 0 or equal to 0 then the output will be 1.

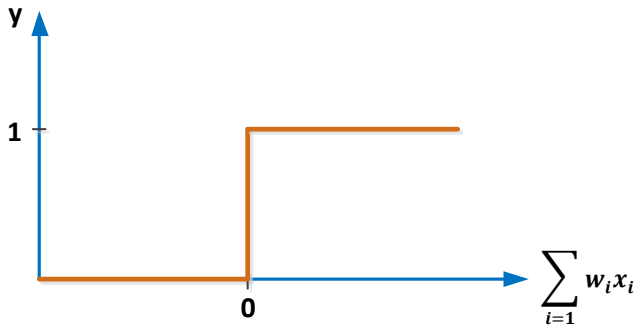



Figure 7. Threshold Function

Mathematical model for the threshold function is given below.

$$\varphi(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

### 3.3.2. Sigmoid Function

As shown in Figure 8, anything below 0 drops off in sigmoid function and above 0, it approximates towards 1. It is very useful in the final layer of the neural network when there is prediction of probabilities. Mathematical formula for the sigmoid function is given below.

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

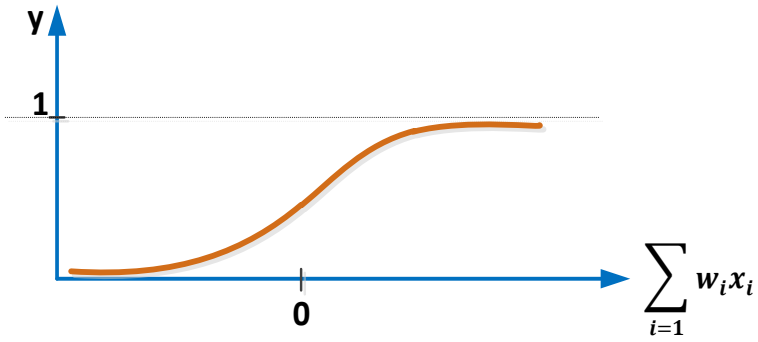



Figure 8. Sigmoid Function

### 3.3.3. Rectifier Function

Rectifier function is one of the most commonly used activation functions. As shown in Figure 9, it goes all the way to 0 until it reaches zero and then from there it gradually progresses as the input value increases as well. Mathematical formula for the rectifier function is given below.

$$\varphi(x) = \max(x, 0)$$

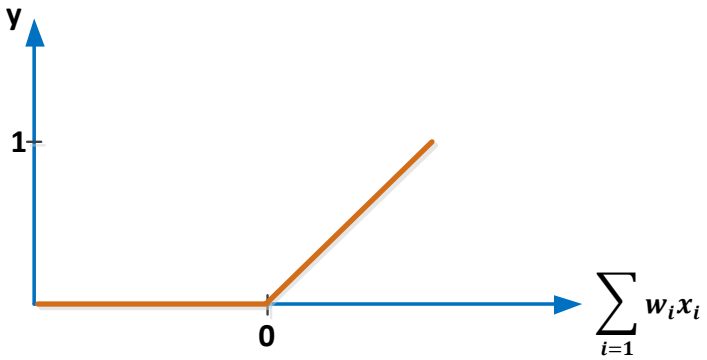



Figure 9. Rectifier Function

### 3.3.4. Hyperbolic Tangent Function

It is very similar to the sigmoid function but the hyperbolic tangent function ($tanh$) has negative values below zero. In Figure 10, values go from zero to one and go from zero to minus one on the other side. Mathematical formula for the hyperbolic tangent function is given below.

$$\varphi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

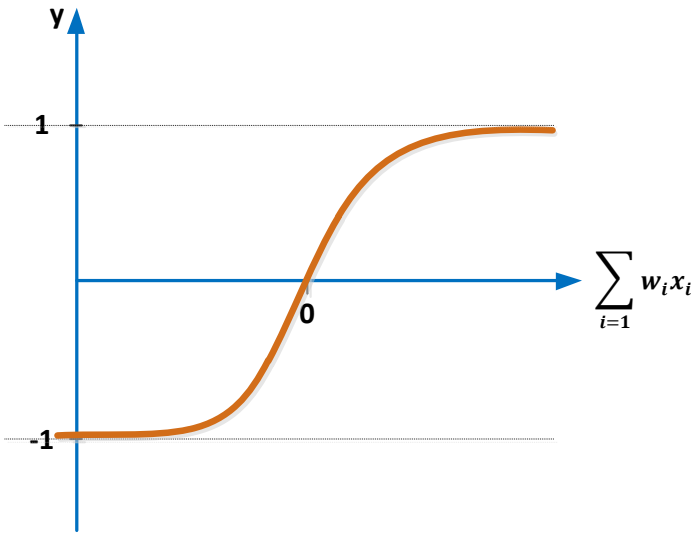



Figure 10. Hyperbolic Tangent Function

## 3.4. Network Architectures

Neural networks can be grouped as feed-forward networks and recurrent networks based on the number of layers and their connection pattern (Jain, Mao, and Mohiuddin, 1996).

### 3.4.1. Feedforward Neural Networks

Networks where the output of one layer is used as the input to the following layer are called feedforward neural networks. These networks are divided as single layer or multi-layer feedforward network. Adding more hidden layers to a network increases its capability to higher-order statistics. Each layer generates output to be used as input in the next layer and final layer generates response of the overall network. Such networks are memory-less because the response to any new input data is not related to network's previous state.

Figure 11 shows the structure of a fully connected feed-forward neural network. A network is referred to as fully-connected in case that each node in a layer is connected to every node in the following layer.

On the other hand, if there are missing links between some of the nodes, network is said to be partially connected (Haykin, 1999).

### 3.4.2. Recurrent Neural Networks

Dynamic networks are called recurrent neural networks. Every time when the outputs are computed, the inputs to each neuron are modified because of the feedback loop, which yields the network to enter a new state. Figure 12 shows the structure of a recurrent neural network.

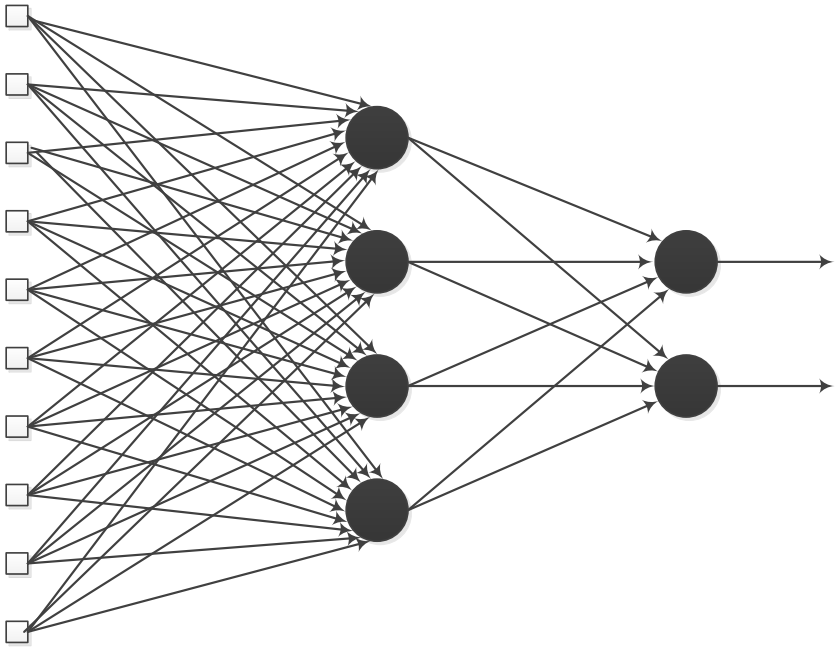


Figure 11. Fully Connected Feed-Forward Neural Network
(Source: Haykin, 1999)

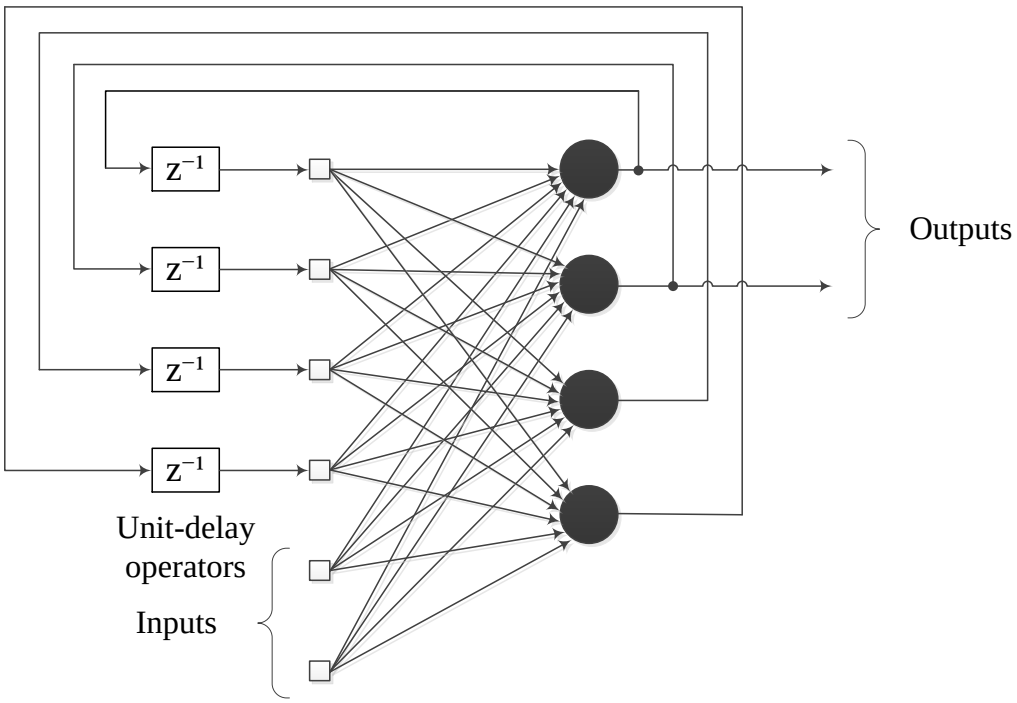



Figure 12. Recurrent Neural Networks with Hidden Neurons
(Source: Haykin, 1999)

## 3.5. Learning Algorithms

Learning algorithms define procedures expressed by prescribed set of rules for tuning network weights. They are categorized as supervised, unsupervised, and hybrid learning algorithms (Jain, Mao, and Mohiuddin, 1996).

### 3.5.1. Supervised Learning

In supervised learning, a network is provided with the desired response for every input. Weights are adjusted in the direction that minimizes difference between actual response and expected outcome. This measurement of learning performance is called as the cost function. Formula given below belongs to the quadratic cost function, which is one of the most commonly used function type, and it is also referred to as the mean squared error.

$$C(w,b) = \frac{1}{2n}\sum_{x}\|y(x) - a\|^2$$

Here, *w* corresponds to the network's weights, *b* is for all the biases, *n* is the number of inputs of a single observation and $\boldsymbol{a}$ is the vector of desired outputs. The purpose of learning algorithms is to minimize *C* as a function of weights and biases.

### Backpropagation Algorithm

Minimizing a function refers to finding where it reaches its global minimum. This can be achieved by using derivatives to find function's extremum points. However, this kind of approach is only useful if such function has one or few variables. In the case of deep neural networks, cost functions depend on tons of weights and biases.

The backpropagation algorithm is one of the most popular learning algorithm that seeks for the minimum of the cost function in weight space using the gradient descent method where gradient $\nabla C$ is repeatedly computed until global minimum is reached.

For any function $C(v)$ where $v = v1, v2, ...,$ the gradient of $C$ relates changes in $v$ and as a result changes in $C$.

$$\Delta C \approx \nabla C \,.\Delta v$$

Assuming that,

$$\Delta v = -\eta \nabla C$$

Here, $\eta$ is the learning rate. Then $\Delta C$ becomes,

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \parallel \nabla C \parallel 2$$

Knowing the fact that $\parallel \nabla C \parallel 2 \geq 0$, $\Delta C$ will always be less than or equal to 0. This means error will always decrease if $v$ is changed according to $\Delta v = -\eta \nabla C$. After $\Delta v$ is computed, $v$ is updated to $v'$ and it is repeated over and over until it reaches a value that results global minimum of function $C$.

$$v' = v - \eta \nabla C$$

This procedure is known as the gradient descent algorithm (Baydar, 2018). Gradient descent algorithm is also referred to as stochastic gradient descent or mini batch gradient or full gradient descent according to the number of samples that fed into the network in each iteration.

In stochastic gradient descent approach, samples are fed to the network one by one and new weights are calculated at every step. However, the disadvantage of this method is that it can stuck on a local minimum instead of global minimum.

By contrast, all training samples are fed into the network and weights are updated at once in full gradient descent method. This method, however, is computationally expensive since all samples are applied to the network together and this can cause memory insufficiency.

The most common method is mini batch gradient descent method. It combines advantages of both methods and only certain number of samples are fed to the network in each iteration. In this way, training time may take longer time than stochastic gradient descent but it can discriminate local minimum problem.

Choosing the right learning rate is also an important issue in gradient descent algorithms. In practical implementations, $\eta$ is often varied (Using Neural Nets to Recognize Handwritten Digits, 2019).

Adaptive Moment Estimation, which is commonly known as the Adam Optimizer, is an improved stochastic gradient descent method that uses adaptive learning rate method. It optimizes a function from the gradient and the squared gradient which are the estimates of first and second moments (Kingma and Ba, 2014).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 V_{t-1} + (1 - \beta_2) {g_t}^2$$

Here, $m_t$ and $v_t$ are moving averages, $g$ is gradient, and $\beta_1 \in [0,1)$ and $\beta_2 \in [0,1)$ are hyper-parameters which control the exponential decay rates of $m_t$ and $v_t$. In most cases, $\beta_1$ is taken as 0.9 and $\beta_2$ is taken as 0.99 (Kingma and Ba, 2014).

In order to update learning step, learning rate is multiplied by average of the gradient then divided by the root mean square of the exponential average of $v_t$ which is the square of gradients. Epsilon ($\varepsilon$) is a very small number and it is used to prevent any division by zero in the implementation.

$$\theta_{t+1} = \theta_t - \frac{\eta \widehat{m}_t}{\sqrt{\widehat{v}_t + \varepsilon}}$$

### 3.5.2. Unsupervised Learning

In unsupervised learning, desired outputs corresponding to each input set are not provided to the network. The weights and biases are determined in response to network

inputs only. Such networks learn to categorize input patterns by examining their data structures or correlations between them (Jain, Mao, and Mohiuddin, 1996).

### 3.5.3. Hybrid Learning

Hybrid learning is a combination of both supervised and unsupervised learning. While majority of the weights are obtained through supervised learning, the rest of them are determined by unsupervised learning.

## 3.6. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a special type of feedforward networks known as the multilayer perceptron that are trained with backpropagation. Main purpose of CNNs is to identify visual patterns directly from images (Kuo, 2016). Inputs of CNNs are processed through several steps referred to as layers. First layer of a CNN is the convolutional layer, which can be followed by layers called ReLU layer, pooling layer, normalization (flattening) layer and fully connected layer.

### 3.6.1. Convolution Layer

A convolution is a mathematical way of combining two continuous-time signals $x(t)$ and $h(t)$ in order to form a third signal $y(t)$ and it is denoted by integration operation as in the formula given below. (Hsu, 2011)

$$(x * h)(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

Convolution is an important operation because the output of any continuous-time linear time-invariant system can be described by the convolution of the input $x(t)$ with the impulse response $h(t)$ of the system.

The convolution operation is also a fundamental component of CNNs. It is applied to the input image by the feature detector, which is often referred to as kernel or filter. Input image can be black and white (B&W) or colored image. B&W images are two-dimensional matrices and each pixel is represented on a scale from 0 to 255 in general. Colored images are three-dimensional and again, each pixel inside a colored image is represented on a scale from 0 to 255 but on three levels as red, green, and blue. Representation of B&W and colored images are given in Figure 13.

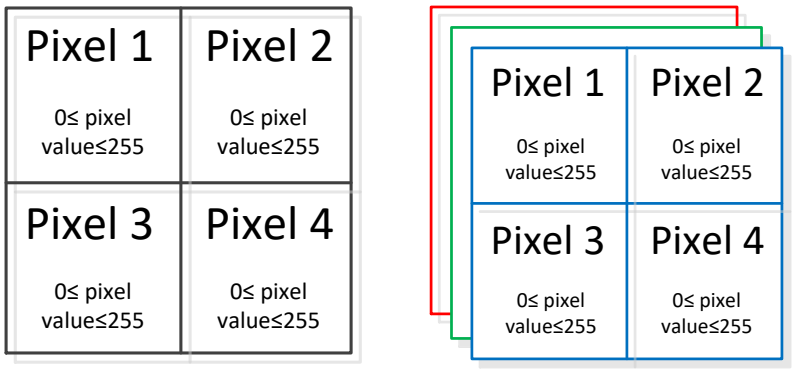



Figure 13. Representation of B&W and Colored Images

On the other hand, filters have small sizes like 3x3 or 5x5 and their depth differs depending on whether input is B&W or colored image.

Figure 14 shows an example of convolution operation in 2D. Initially, filter is placed on top left of the input matrix, then the number of matching cells are counted. This number is inserted into a new matrix called the feature map. As feature detector is shifted to the right and moved down until it reaches the downright border of the input matrix, number of matching cells are counted and inserted into the feature map respectively. Collection of these feature maps that are developed by multiple feature detectors are referred to as convolutional layers. Feature detectors are useful for reducing the size of

the input image. They eliminate the unnecessary parts in the image while still emphasizing certain features.

Another important concept in convolution is the stride; *S*. It specifies the number of steps to shift the feature detector at each movement. If $s > 1$, feature detector will skip $s - 1$ pixel in every shift and the convolution will be performed once every *s* pixels (Wu, 2017). Output size of a convolutional layer can be expressed by:

$$Output\ Size = \frac{W - K - 2P}{S} + 1$$

Here *W* is the size of the input, *K* is size of kernel and *P* and *S* are the padding value and stride respectively.

### 3.6.2. ReLU Layer

The rectified linear unit (ReLU) is not a separate component; it is more like a supplementary step to the convolution operation. Rectifier function is linear for all positive values, and zero for all negative values. Therefore it converts each negative pixel into zero in a feature map and keeps each positive pixel (Wu, 2017).

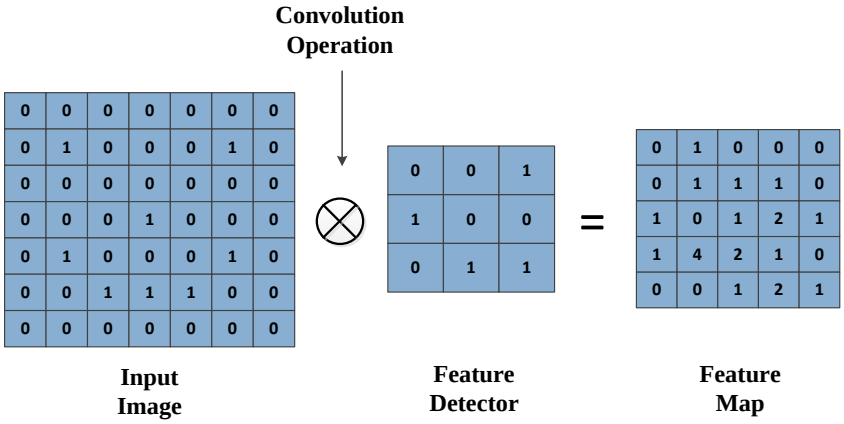



Figure 14. Example of Convolution Operation
(Source: The Ultimate Guide to Convolutional Neural Networks (CNN), 2018))

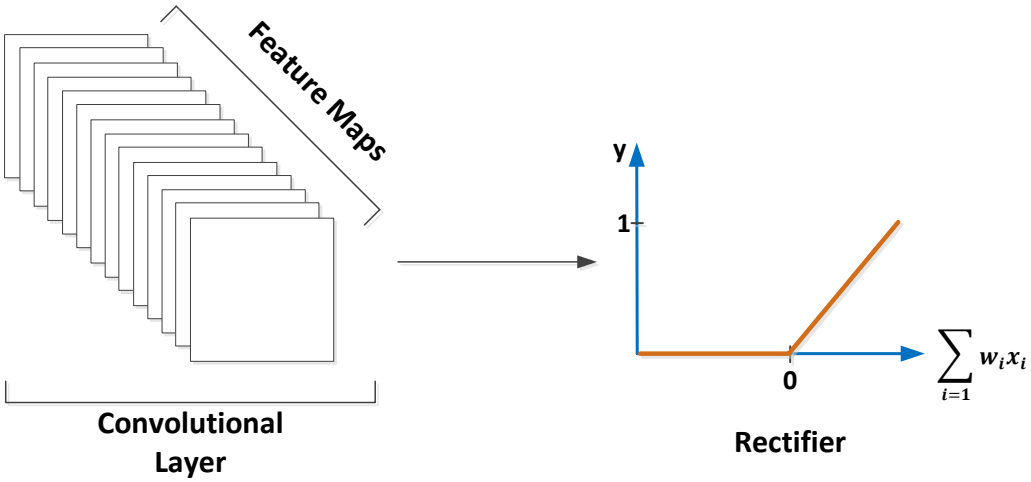



Figure 15. Rectifier Function as Activation Function

There are many non-linear features in an image e.g. the transition between pixels, borders, colors, etc. Since convolution is a linear operation, ReLU is often preferred between the convolutional layers in order to increase non-linearity because it is the simplest non-linear function and it's efficient for computation.

### 3.6.3. Pooling Layer

Pooling process provides network a capability to detect specific patterns within the image. Features obtained from convolutional layer can be high in number and training on too many features can cause overfitting. Therefore, a pooling process is usually applied to reduce dimensions. There are different types of pooling operators such as mean pooling, sum pooling and max pooling which is the most commonly used operator. In max pooling, the pooling operator is shifted throughout the entire input image similar to a feature detector however, instead of summing matching pixels; it takes the maximum value of the sub region in each shift.

The purpose of pooling layers is to dispose of unnecessary information or features in an image for a simpler output. It provides the convolutional neural network with the spatial variance capability (Stenroos, 2017).

Figure 16 shows an example of max pooling operation where pooling size is 3x3 and stride is 1.

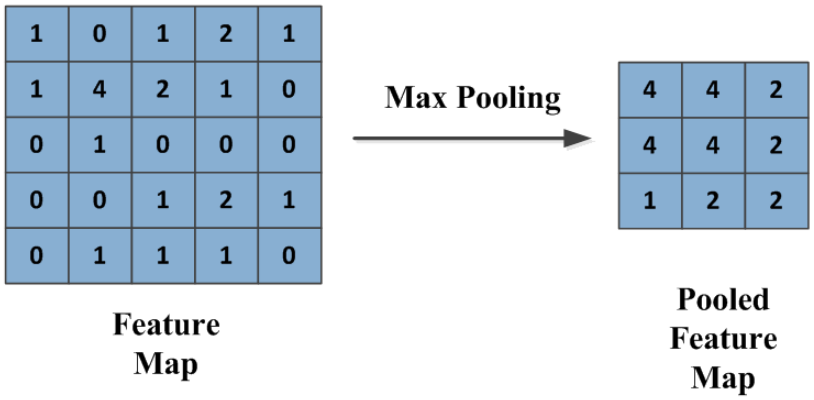



Figure 16. Example of Max Pooling Operation

### 3.6.4. Flattening Layer

In this step, pooled feature maps are flattened into a vector of input data to be passed through the fully connected layer. Figure 17 shows an example of the flattening operation.

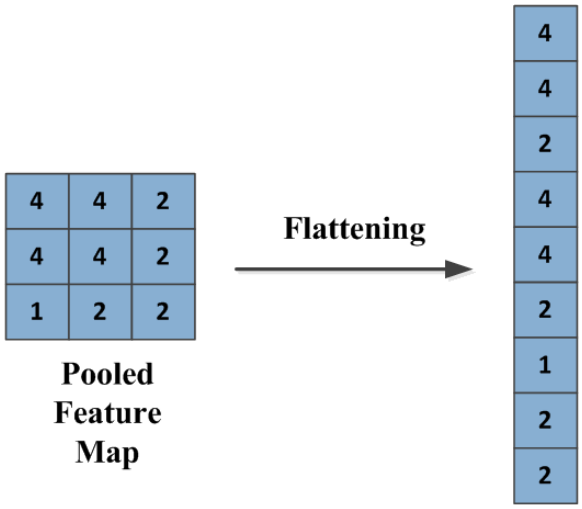



Figure 17. Example of Flattening Operation

### 3.6.5. Fully Connected Layer

A fully connected layer refers to a layer where convolutional neural network is evolved into an artificial neural network. All neurons are connected to this layer. By the end of this layer, network outcomes a prediction, then cost function, which is referred to as loss function in convolutional neural networks, is calculated.

Loss function is the measure of how accurate the network is and in order to increase effectiveness of the network, weights and the feature detectors are adjusted as discussed in Section 3.4.1.

Generally, a special activation function is used at the last fully connected layer, which will generate a probabilistic result for each class. For multi-class problems, mostly Softmax function is preferred.

#### Softmax Function

Softmax function calculates the probability of each class. In other words, it shows which class the input is more likely to match and since it outputs the probabilities, the sum of all values equals to 1. Softmax function can be expressed as:

$$softmax(x_k) = \frac{e^{x_k}}{\sum_{i=1}^{C} e^{x_i}}$$

Here, $C$ is the number of classes.

### 3.6.6. Dropout

Overfitting is one of the most common problems in convolutional neural networks and it occurs when the network memorizes the training set. In order to prevent network

from overfitting, dropout method randomly eliminates a certain percentage of the neurons intentionally during training by setting activations to 0. This makes the system not too dependent on a single neuron or connection. Generally, dropout is applied in the fully connected layers of convolutional neural networks. Figure 18 shows an example of dropout operation.

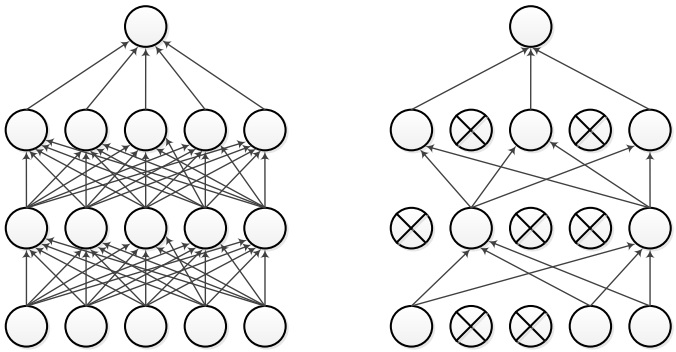


Figure 18. Example of Dropout Operation
(Source: Dropout Neural Network Layer In Keras Explained, 2019))

### 3.6.7. Data Augmentation

Data augmentation is another method for preventing overfitting. When it is not possible to acquire more training data, data augmentation is used to artificially increase the size of the training set by producing data by altering actual inputs. The most common data augmentation methods are flipping, scaling, rotation etc.

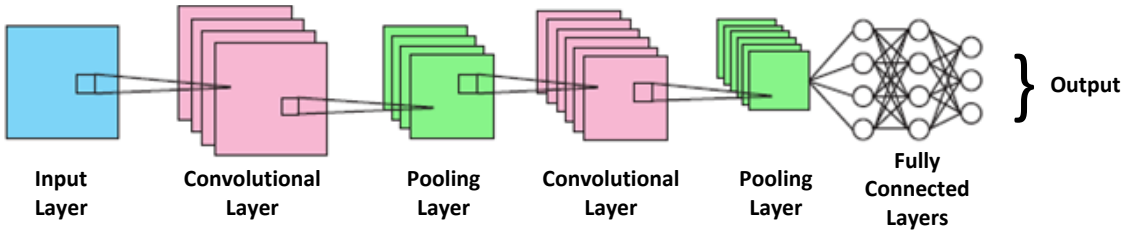



Figure 19. Example of a Convolutional Neural Network
(Source: Stenroos, 2017)

In a typical convolutional neural network, layers explained above are stacked consecutively as shown in Figure 19. For example, the outputs of the first convolution-pooling layer can be used as an input to a similar convolution-pooling layer. This multi-layer approach increases complexity of the network as it can learn more and more complicated structures through higher layers.

# CHAPTER 4

# EXPERIMENTAL WORK AND RESULTS

Experimental work of this study is divided into several steps as shown in Figure 20 and it can be summarized as follows in general:

Ottoman alphabet character dataset was created in the first two steps. In the third step, a deep convolutional network model was created and the model was trained with the dataset prepared. Then, by modifying the layers, effects of network complexity to the recognition results were examined.

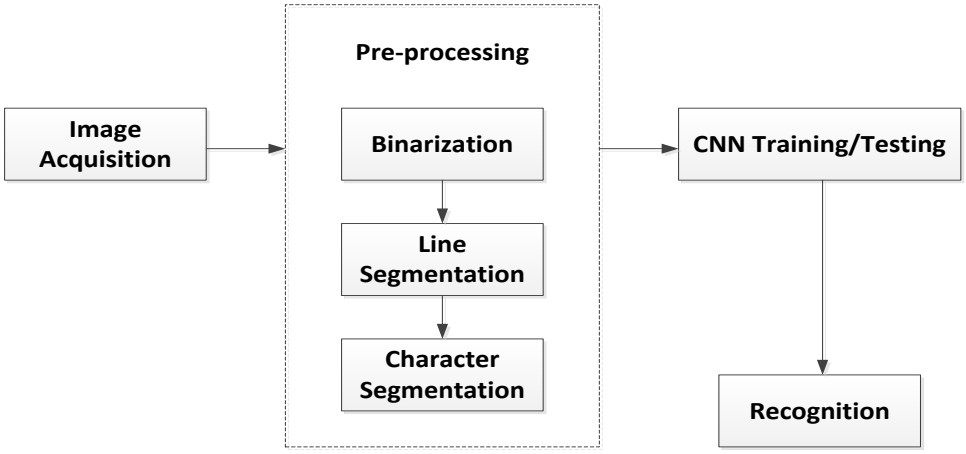



Figure 20. Architecture of the Experimental Work

## 4.1. Creating Dataset

Since Ottoman alphabet has lost its validity, it is not possible to find a ready Ottoman alphabet character dataset easily today. Therefore, in this study, scanned samples of various Ottoman literature works were collected from the digital database of

Turkish National Library for creating dataset by randomly selecting texts of different lengths from selected documents. Collected documents are machine printed and non-text contents were avoided during selection. 130 images of different lengths were selected. Figure 21 shows one of the selected documents as an example.

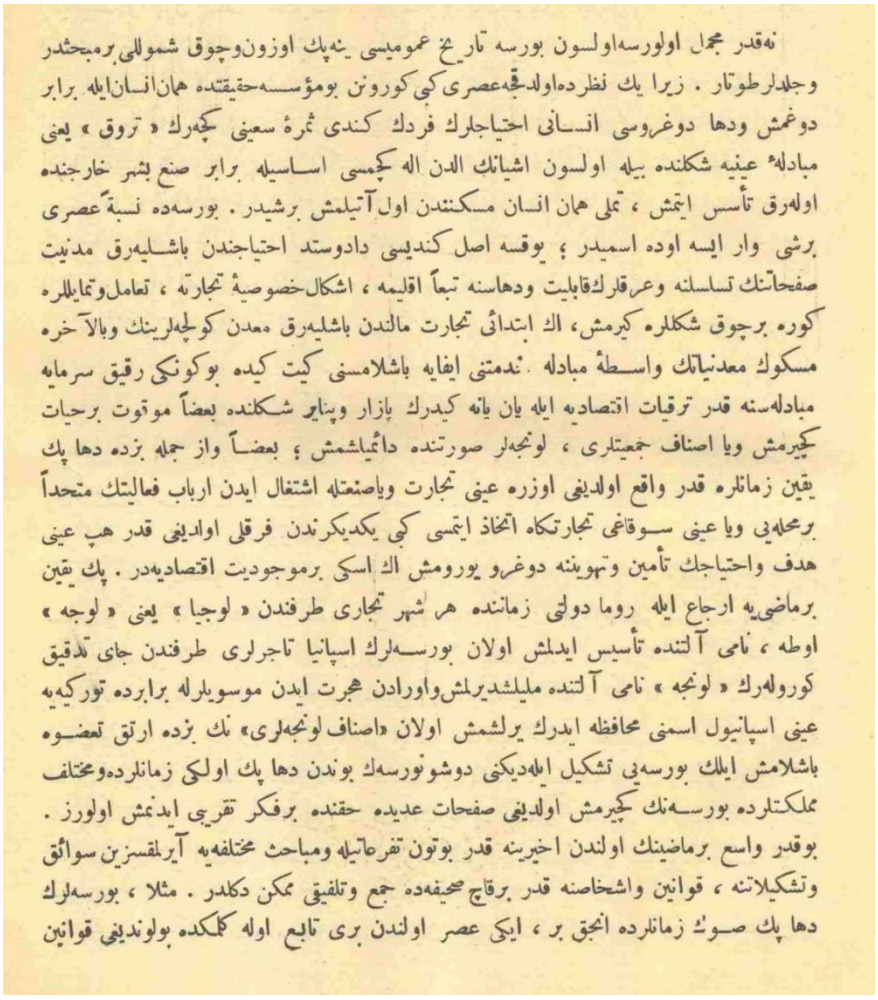


Figure 21. Example of Ottoman Text
(Source: Sırat-ı Müstakim, 1908)

Binarization is the initial step for the segmentation. At first, all images were converted to the grayscale images. Figure 22 shows the grayscale version of the above text.

In order to convert pixel values from shades of gray, to 1's and 0's, a threshold value is used.

Threshold binarization method can be described as:

$$g(x,y) = \begin{cases} 1, if\ f(x,y) \geq T \\ 0, otherwise \end{cases}$$

where $T$ is the global threshold value (Converting a Grayscale Image to Binary Image Using Thresholding, n.d.).

Figure 23 shows the binary image of Figure 22.

Figure 22. Same Text in Grayscale Form

Segmentation of the text images into lines, words and finally to characters is one of the most critical steps in an OCR system. Texts that are written in Latin alphabet are interconnected only if they are handwritten, however the Ottoman documents contain interconnected scripts even in machine printed texts. Therefore, segmentation of each character in an Ottoman script is a challenging problem and any segmentation error affects overall system recognition success adversely.

Figure 23. Binarized Form of the Text

In this study, segmentation is performed in two sections as line segmentation and character segmentation.

### 4.1.1. Line Segmentation

In order to segment lines from the binary image, horizontal projections are created first, by summing pixels of each row. Horizontal projection of Figure 23 is shown below.

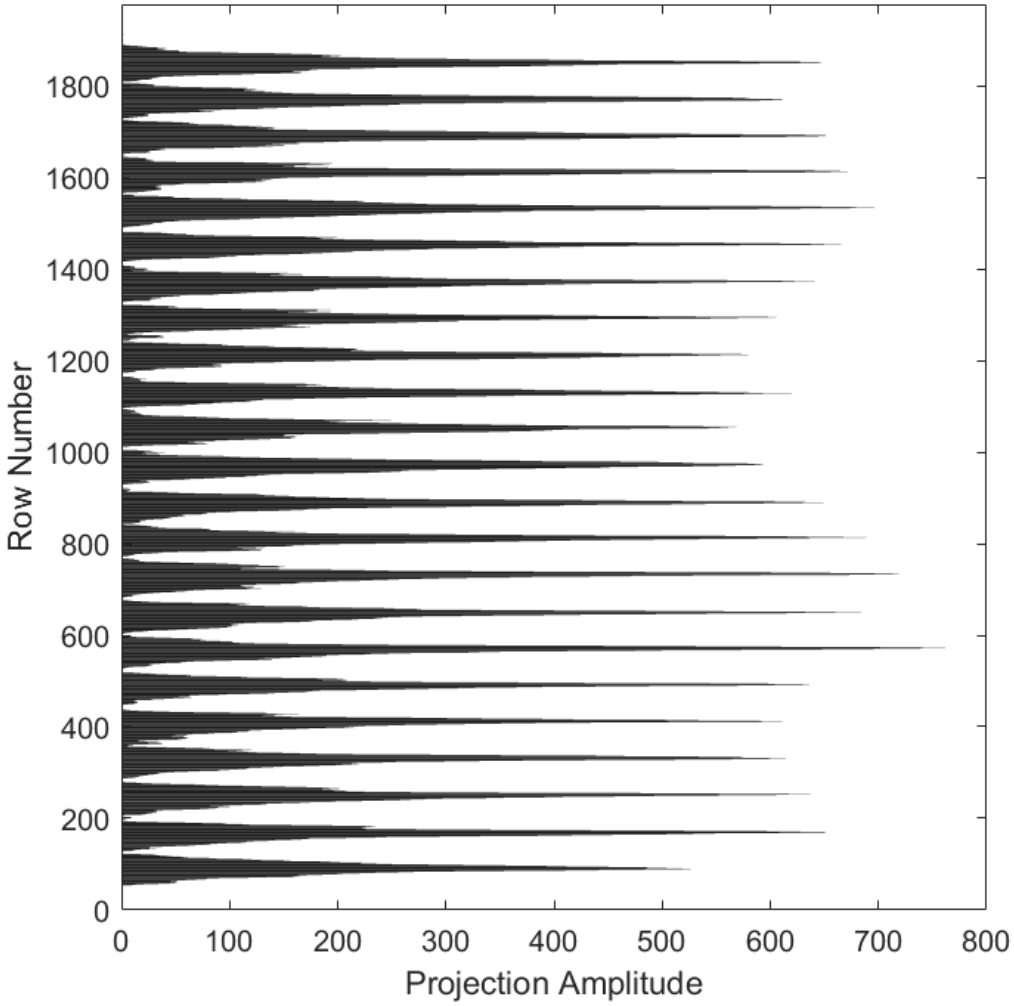



Figure 24. Horizontal Projection of Figure 23

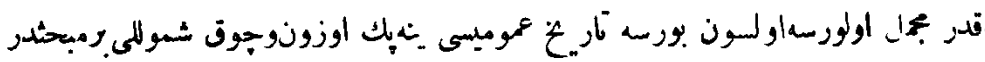
If the white pixels are expressed as 0, and black pixels are expressed as 1 in the binary image, higher peaks show that, numbers of black pixels along that row are high and numbers of white pixels along that row are low. According to these peak points, it can be seen from the Figure 24, the text shown in Figure 23 has 23 lines.

Assume that $B\,[i,j]$ is a binary image whose horizontal projection can be described as:

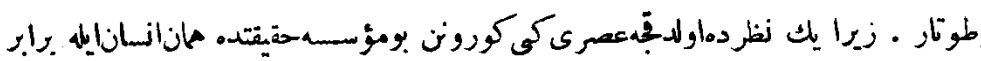
$$H[i] = \sum_{j} B[i,j]$$

$$L[i] = \begin{cases} 1, if\ H[i] >= T) \\ 0, \quad otherwise \end{cases}$$

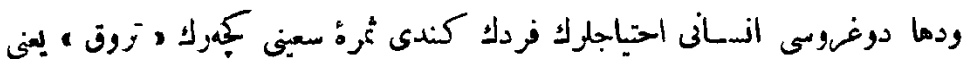
Each row that contains text is categorized as 1 and each non-text row is categorized as 0 by comparing $H[i]$ with a threshold $T$. Then, the transitions from 0 to 1 and transitions from 1 to 0 of the value of each element of $L\,[i]$ are marked. As a result, lines are extracted from the original image according to marked rows.



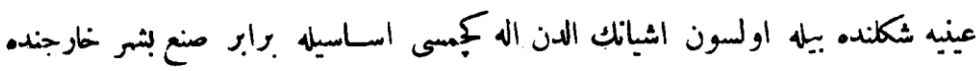
(a) Line 1



(b) Line 2



(c) Line 3



(d) Line 4

Figure 25. (cont. on next page)

اولەرق تأسس ايتمش ، تملى همان انسان مسكنندن اول آتيلمش برشيدر . بورسەده نسبةً عصرى

(e) Line 5

برشى وار ايسه اوده اسميدر ؛ يوقسه اصل كنديسى دادوستد احتياجندن باشليەرق مدنيت

(f) Line 6

صفحاتنك تسلسلنه وعرقلرك قابليت ودهاسنه تبعاً اقليمه ، اشكال خصوصيهٔ تجارتە ، تعامل وتمايللرە

(g) Line 7

كوره برچوق شكللرە كيرمش، الك ابتدائى تجارت مالندن باشليەرق معدن كولچەلرينك وبالآخرە

(h) Line 8

مسكوك معدنياتك واسطهٔ مبادله ندمتنى ايفايه باشلامسنى كيت كيده بوكونكى رقيق سرمايه

(i) Line 9

مبادلەسنه قدر ترقيات اقتصاديه ايله يان يانه كيدرك پازار وپنايرِ شكلنده بعضاً موقوت برحيات

(j) Line 10

كچيرمش ويا اصناف جمعيتلرى ، لونجەلر صورتنده دائميلشمش ؛ بعضاً واز جمله بزده دها پك

(k) Line 11

Figure 25. (cont. on next page)

برمحلهيي ويا عينى سوقاغى تجارتكاه اتخاذ ايتمسى كبى يكديكرندن فرقلى اولديغى قدر هپ عينى

(m) Line 13

هدف واحتياجك تامين وتهوينه دوغرو يورومش اك اسكى برموجوديت اقتصاديهدر . پك يقين

(n) Line 14

برماضيه ارجاع ايله روما دولتى زمانده هر شهر تجارى طرفندن « لوجيا » يعنى « لوجه »

(o) Line 15

اوطه ، نامى آلتنده تأسيس ايدلمش اولان بورسهلرك اسپانيا تاجرلرى طرفندن جاى تدقيق

(p) Line 16

كورولهرك « لونجه » نامى آلتنده مليلشديرلمش واورادن هجرت ايدن موسويلرله برابرده توركيهيه

(q) Line 17

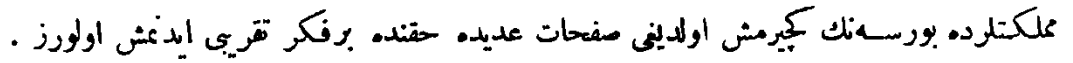
عينى اسپانيول اسمنى محافظه ايدرك يرلشمش اولان «اصناف لونجهلرى» نك بزده ارتق تعضوه

(r) Line 18

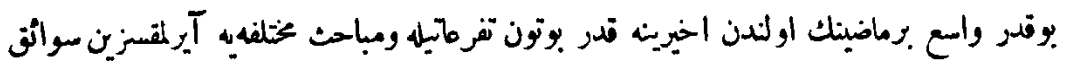
باشلامش ايلك بورسهيى تشكيل ايلهديكنى دوشونورسهك بوندن دها پك اولكى زمانلردهوختلف

(s) Line 19

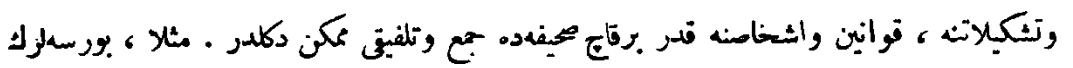
Figure 25. (cont. on next page)



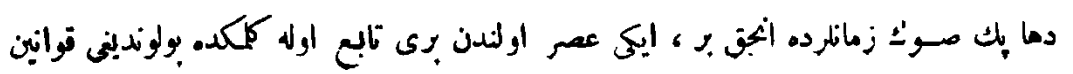
(t) Line 20



(u) Line 21



(w) Line 22



(v) Line 23

Figure 25. Lines Extracted from the Text in Figure 23 (cont.)

### 4.1.2. Character Segmentation

After line segmentation, character segmentation is applied to the extracted lines.

However, due to the characteristics of the Ottoman alphabet, some characters are written interconnected even if the texts are machine printed. Therefore, character segmentation is not straightforward as it is in line segmentation.

At first, by applying the same method used in line segmentation but using vertical projections this time, a line is segmented into the isolated characters and interconnected character groups. Vertical projection can be described as:

$$V[j] = \sum_{j} B[i,j]$$

Figure 26 shows the projection of Figure 25 (a), Figure 27 and Figure 28 shows some of the isolated and interconnected character groups that are segmented by vertical projection.

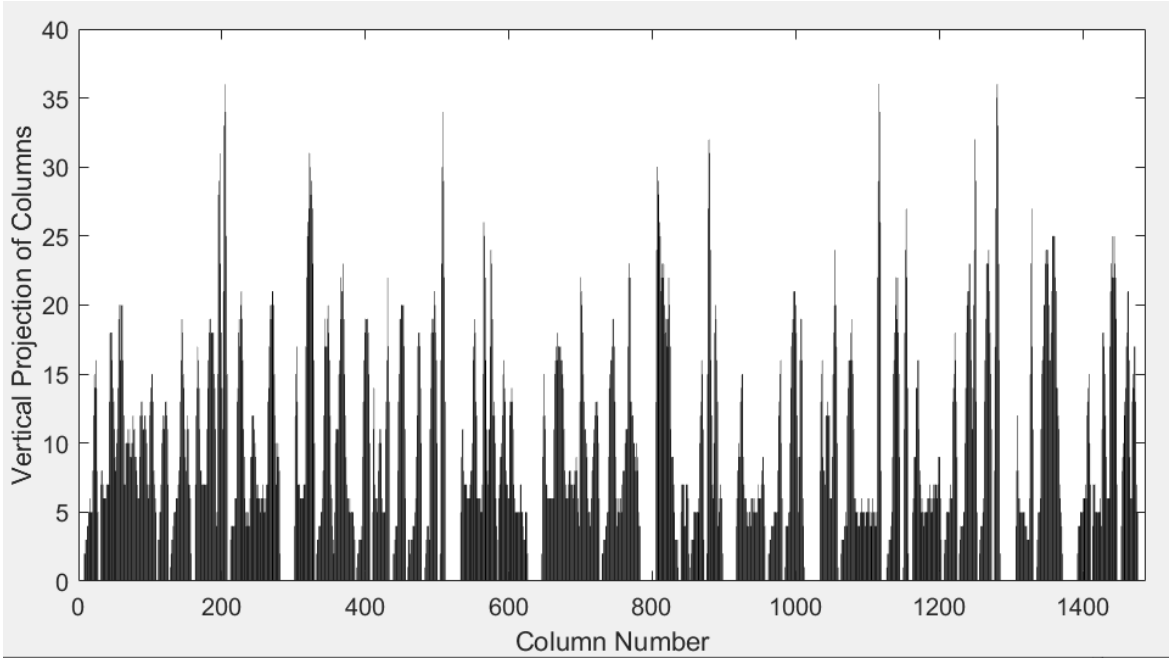



Figure 26. Vertical Projection of Line 1 in Figure 25

Figure 27. Some of the Isolated Characters of Line 1

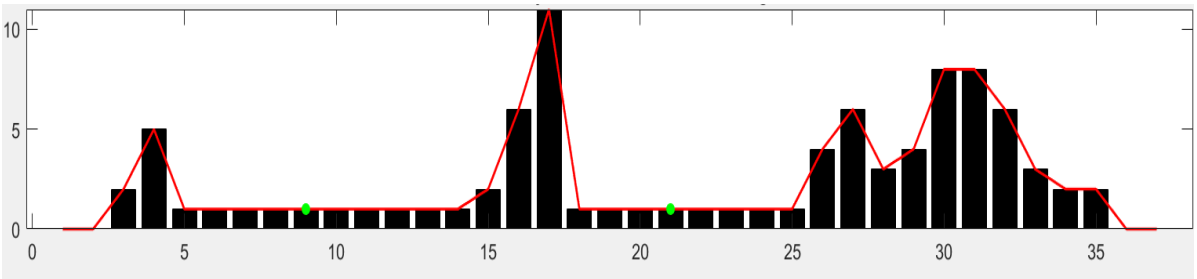
Figure 28. Some of the Interconnected Characters of Line 1

In order to segment interconnected characters, skeletonization is applied to the binary image first. The purpose of this step is to extract the shape feature representing the general form of the characters.

As it is seen from the skeletonized image in Figure 29 (b), two characters are connected with a thin line and this line corresponds to the columns whose values are 1 in the vertical projection shown in Figure 29 (c).

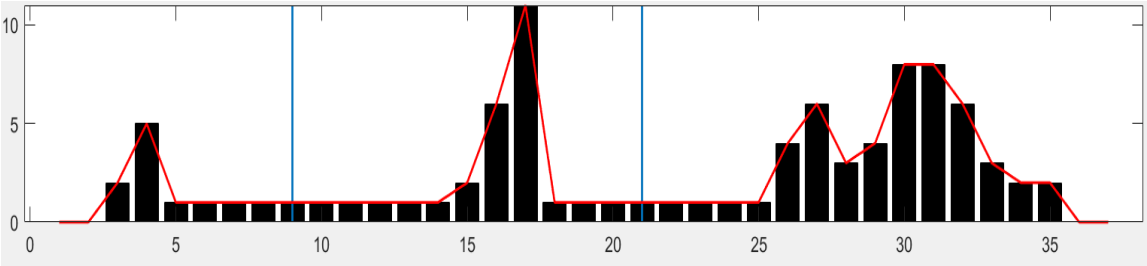
Then, intersections of the columns whose values are 1 to the local minimum points of the vertical projection plot are found. Connection point of two characters in the image will correspond to one of these intersection points. Therefore, the image is divided into parts from the intersection points then the meaningful segmentation is selected. As an example, a connected character set shown in Figure 29 is segmented out from the second intersection point as shown in Figure 30.

(a) (b)




(c)

Figure 29. Segmentation Example of Interconnected Characters (a) Original Image (b) Skeletonized Image (c) Vertical Projection of Skeletonized Image




(a)

(b)

(c)

(d)

Figure 30. Segmentation of Interconnected Characters (a) Vertical Projection of Skeletonized Image (b) Original Image (c) Segmented Character 1 (d) Segmented Character 2

### 4.1.3. Ottoman Alphabet Dataset

By using methods explained in section 4.1.1 and 4.1.2, Ottoman texts that are collected from the database of Turkish National Library have been segmented into 24 categories. For each category, 400 character images were obtained and a dataset is made

of 9600 images in total. Table 3 shows examples of each segmented character in the dataset.

Table 3. Characters Distributed to 24 Classes for the Presented Dataset

| | | | | |
|---|---|---|---|---|
|  |  |  |  |  |
|  |  | | | |
|  |  |  |  | |
|  |  | | | |
|  | | | | |
|  |  |  | | |
|  |  |  | | |
|  |  |  | | |

(cont. on next page)

Table 3. (cont.)

| | | | |
|---|---|---|---|
| ق | ڤ | ڧ | |
| ك | ڪ | | |
| ل | | | |
| لا | | | |
| م | ۾ | ۿ | |
| ن | | | |
| ر | | | |
| ص | ض | ص | ض |
| س | س | | |

(cont. on next page)

Table 3. (cont.)

| | |
|---|---|
| ش | ش |
| ت | ت |
| ط | ظ |
| و | |
| ی | |
| ز | |
| ذ | ذ |

## 4.2. Building Deep Convolutional Neural Network Model

Similar to libraries in computer programming, a framework contains predefined functions for specific tasks. There are many frameworks in the field of deep learning such as Pytorch, Theano, and Tensorflow. In this work, a high-level neural network API called

Keras was preferred for the implementation. Keras is an open source library that runs on top of Tensorflow framework (Keras: The Python Deep Learning Library, 2019).

The time required to train a deep convolutional network may vary depending on the size of the dataset and available processing power. CPU based computation or GPU based computation can be chosen depending on the needs of the application.

CPU based computation options are simpler and more available but it takes longer time to train a network because tasks are computed in serial configuration in a CPU. By contrast, GPU based computation is a time saving option since tasks are handled in parallel configuration. A CUDA based NVIDIA GPU with a minimum of 3.0 computation capability is required for this purpose.

In this study, a desktop computer with an AMD FX-8320 Eight-Core CPU, 8 GBs of RAM, NVIDIA GeForce GTX 760 GPU and Windows 10 operating system was used as an implementation setup. Graphical processer used in this study is CUDA-enabled and it has 3.0 computing capability (CUDA GPUs Computing Capabilities, 2019). Hence, implementation is carried out with GPU version of Tensorflow.

Furthermore, Anaconda Spyder (3.3.3), the scientific python development environment was used for programming purposes.

To measure the network performance, dataset was split into train and test sub datasets. The training set is used to train the model with the known output and purpose of test set is to check the final model performance after training. For splitting dataset, 80% to 20% is commonly used ratio and often referred to as Pareto principle. This means that 80% of the data is used to train the network, and 20% to validate the network.

In this study, in order to classify Ottoman alphabet, dataset was split by 80% to 20% at first, that is, 320 images were used for training and 80 images were used for testing for each character type. In addition, model performance is also tested again with the same dataset divided by 75% for training and 25% for testing. Image selection was made randomly in both experiments.

The best approach to define a deep neural network configuration is to trial and error method since there is no rule of thumb for correct layer configuration. Since the main purpose of this study is to classify the given input into 24 classes, different layer configurations were evaluated with the created dataset in order to achieve better test results.

### 4.2.1. CNN Model-I

Initially, a simple convolutional neural network was built which has a convolution layer with a filter size of 3x3 in the first layer. This layer creates 32 feature detectors that is convolved with the input layers to produce a tensor of outputs. In this case, input layer is binarized character images.

Then, each output in the convolution layer were activated by using the rectified linear unit as an activation function.

The output of this layer was embedded with the max pooling layer that has a pool size of 2x2. Finally, pooled feature maps were flattened and the fully connected layer of 128 neurons at the end classified the given input to one of 24 neurons of the output layer. Layer structure of the proposed model is shown in Figure 31.

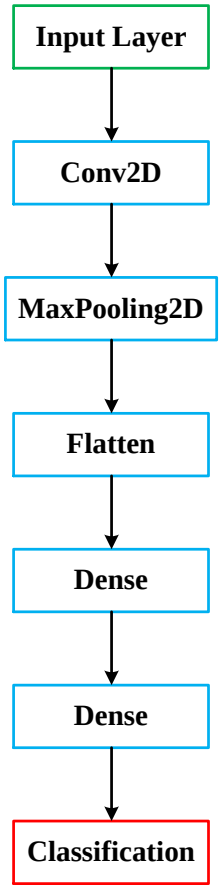



Figure 31. Layer Structure of the CNN Model-I

Softmax activation function was used to output probabilities of each class in order to represent which class is more likely to match to the given input. Because of this is a multi-class problem, categorical cross-entropy was chosen as a cost function for updating the weights during training.

Adam optimizer with adaptive learning rate was used in order to find the minimum point of the cost function.

Network was trained by using 7680 images while data set was split by 80% to 20%. Epoch number was selected 25 while batch size was 32. This means weights were updated in every 32 samples and this process was repeated 25 times.

As can be seen in Figure 32, accuracy of test set was reached to 84.84% while accuracy of training set was reached to 94.93%. On the other hand, validation loss decreased to 64.74% as the model learned.

Same model was trained and evaluated again by splitting dataset as %75 training data and %25 test data. 7200 images were used in training and trained model was tested with 2400 images.

Training and test results were shown in Figure 34 and 35. Accuracy of 85.25% was obtained on the test set while loss was dropped to 53.62% at the end of the training.

Table 4. Architecture of the CNN Model-I

| Layer No (Type) | Output Shape | Parameter |
|---|---|---|
| Conv2D | 148, 148, 32 | 320 |
| MaxPooling2D | 74, 74, 32 | 0 |
| Flatten | 175232 | 0 |
| Dense | 128 | 22429824 |
| Dense | 24 | 3096 |
| Total Parameters: 22433240<br>Trainable Parameters: 22433240<br>Non-trainable Parameters: 0 | | |

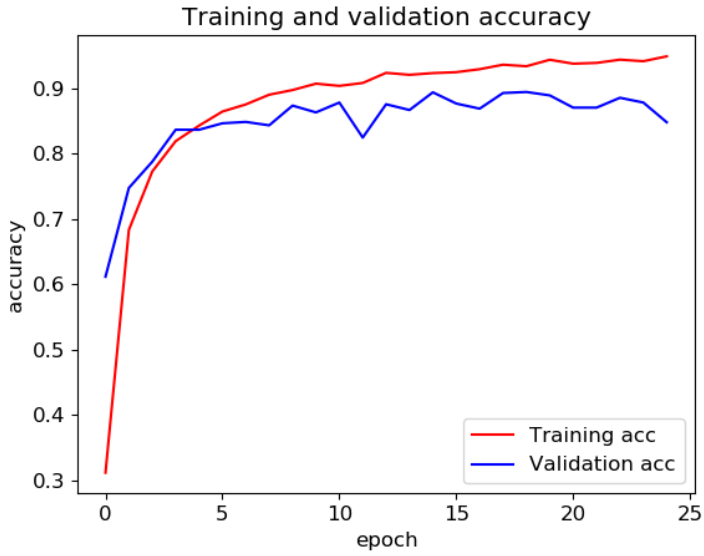



Figure 32. Accuracy Results of the CNN Model-I (Dataset Separated by 80% to 20%)

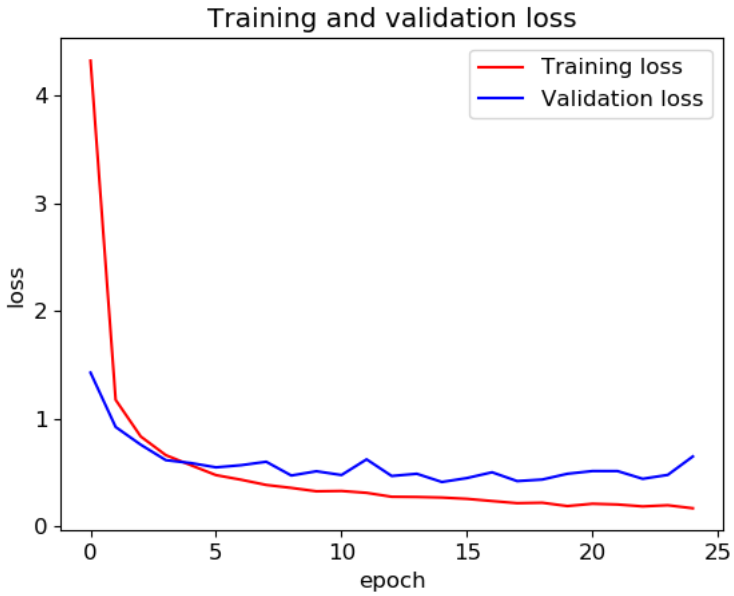



Figure 33. Loss Results of the CNN Model-I (Dataset Separated by 80% to 20%)

Then, number of epochs was increased to 100 and model was trained and evaluated again by using the same dataset. As seen in Figure 36 and 37, model accuracies and losses were reached to 94.46% and 34.58% respectively.

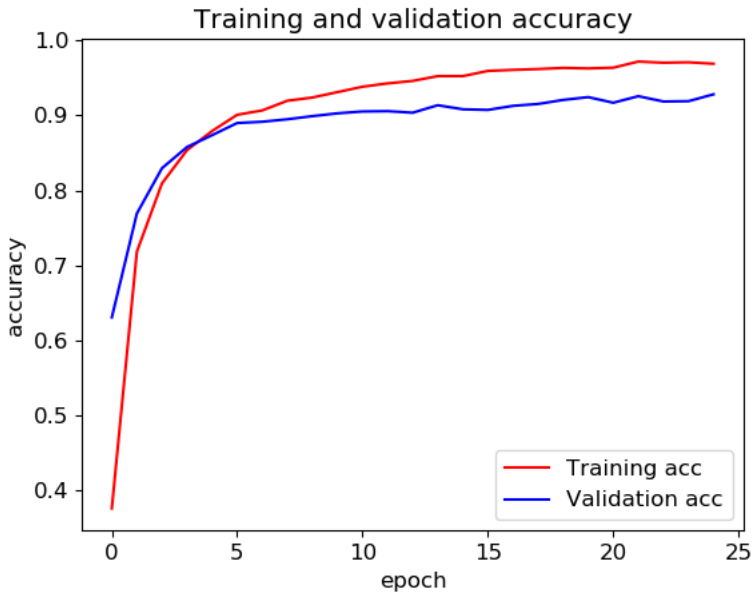



Figure 34. Accuracy Results of the CNN Model-I (Dataset Separated by 75% to 25%)

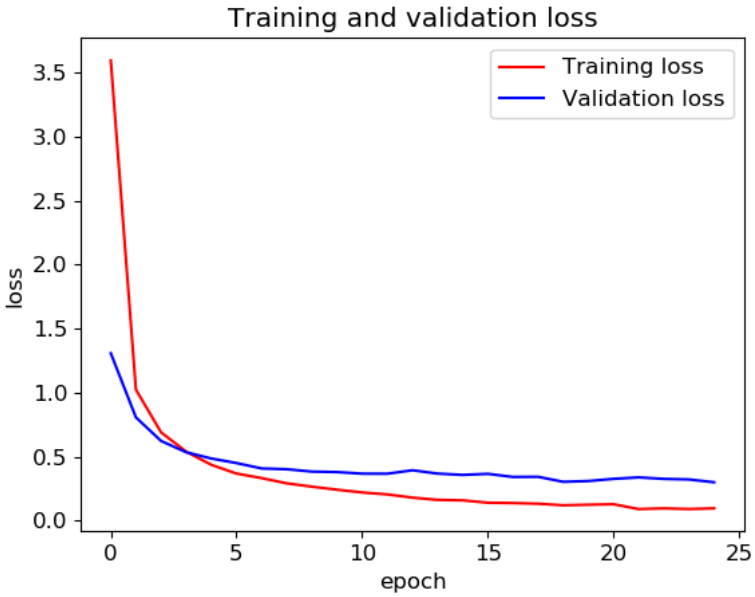



Figure 35. Loss Results of the CNN Model-I (Dataset Separated by 75% to 25%)


Figure 36. Accuracy Results of the CNN Model-I (Dataset Separated by 75% to 25%, Epoch=100)

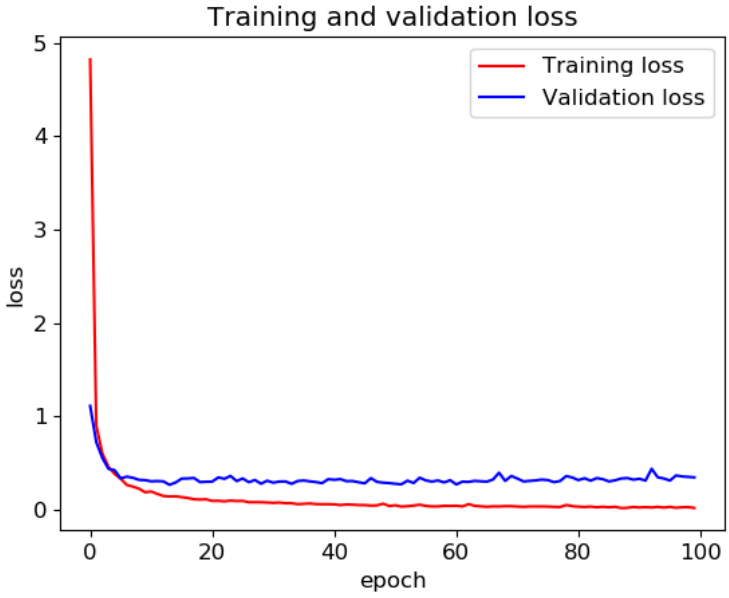



Figure 37. Loss Results of the Proposed CNN (Dataset Separated by 75% to 25%, Epoch=100)

In order to measure the performance and evaluate misclassified data confusion matrices were created for every case. The diagonal cells in Figures 38, 39 and 40 show the number of correct classifications of each individual classes, which are the characters of Ottoman alphabet in this case. Moreover, values in the non-diagonal cells indicate the number of incorrectly classified observations.

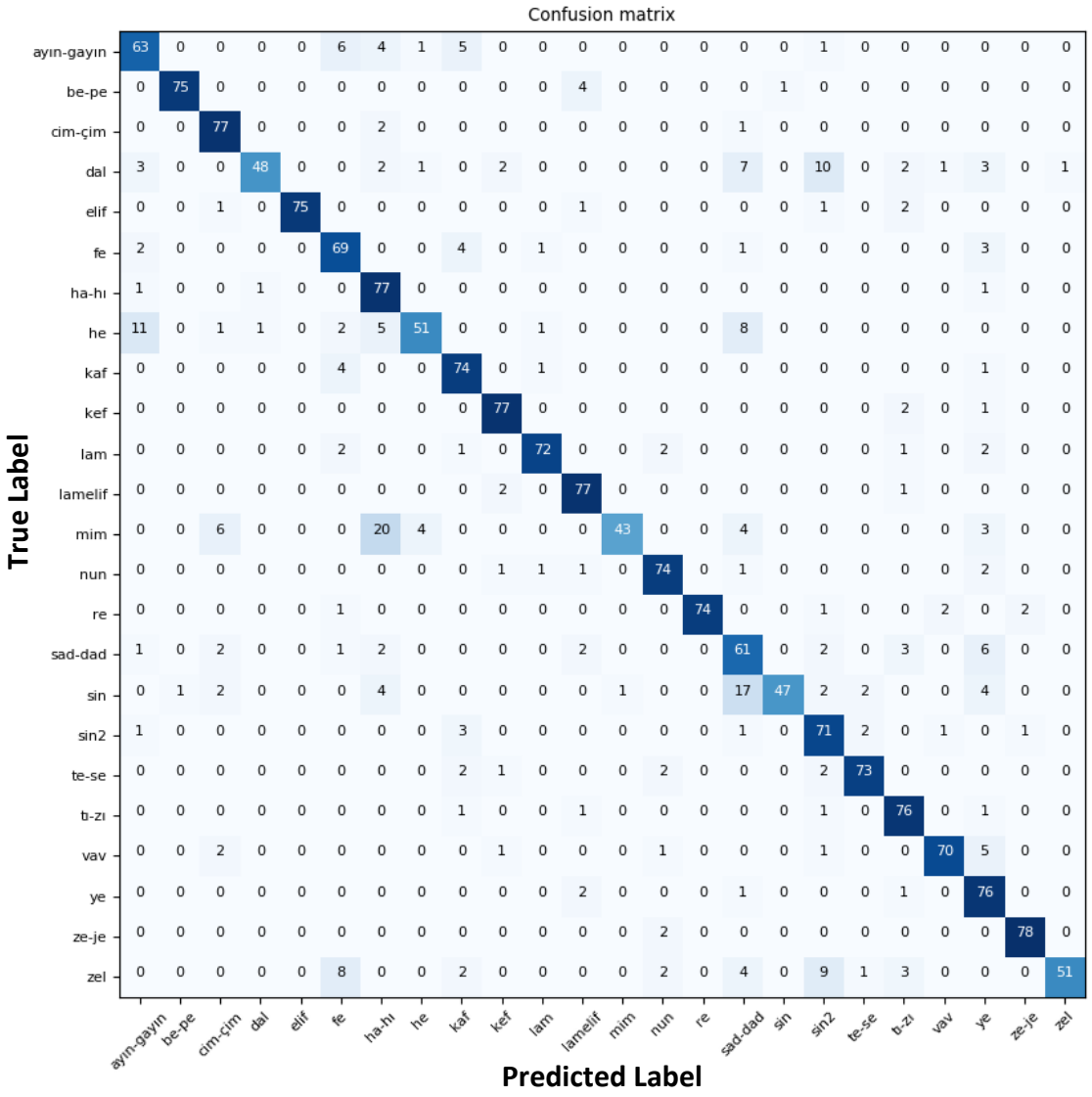



Figure 38. Confusion Matrix of the CNN Model-I (Dataset Separated by 80% to 20%)

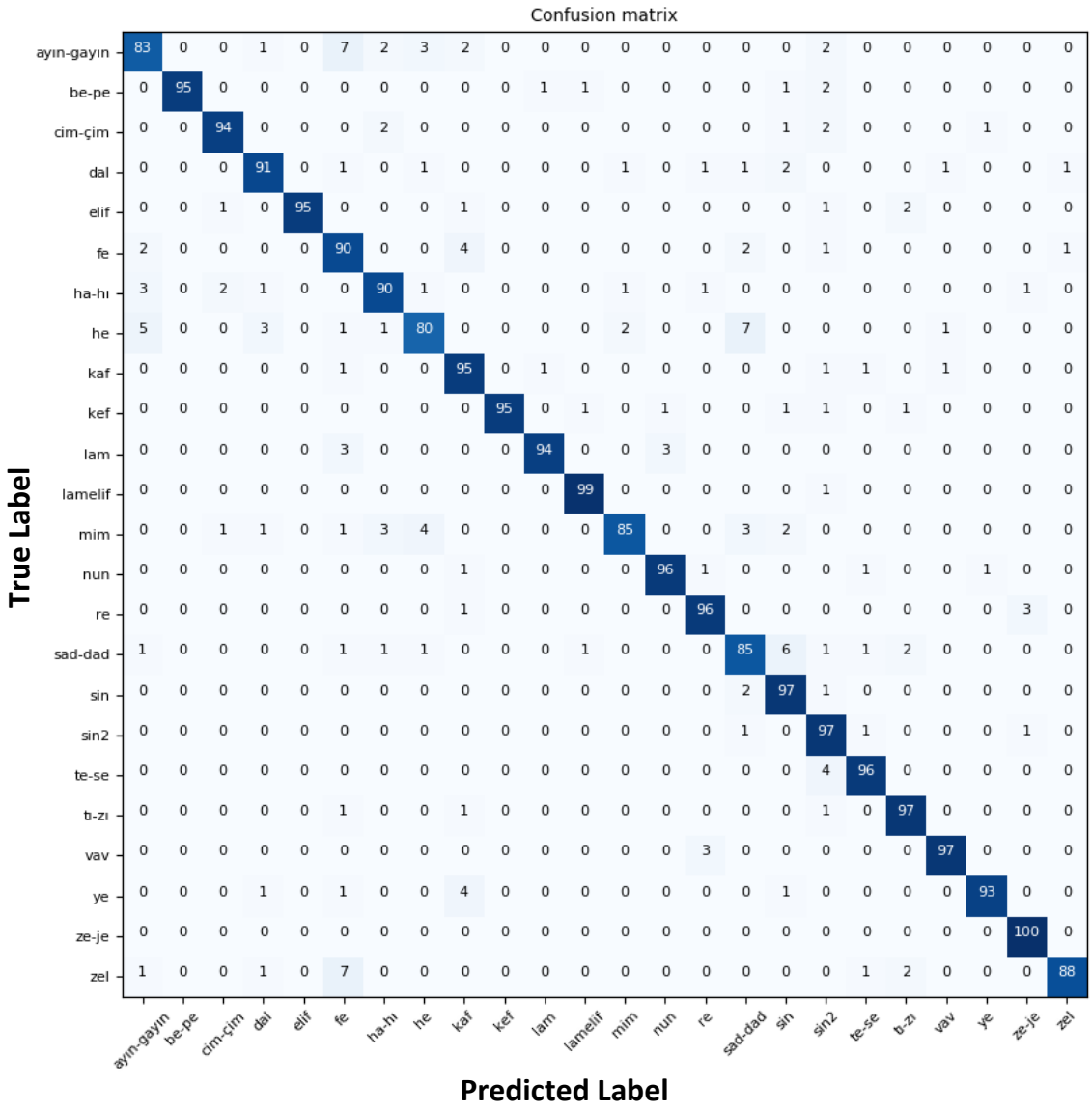



Figure 39. Confusion Matrix of the CNN Model-I (Dataset Separated by 75% to 25%)

## 4.2.2. CNN Model-II

In order to achieve better accuracy and loss performances, more complex network was built by adding more hidden layers this time. Similar to the previous network, layer 1 and 2 were also a convolutional layer with a filter size of 3x3 and they are followed by max pooling layer that has a pool size of 2x2 respectively.

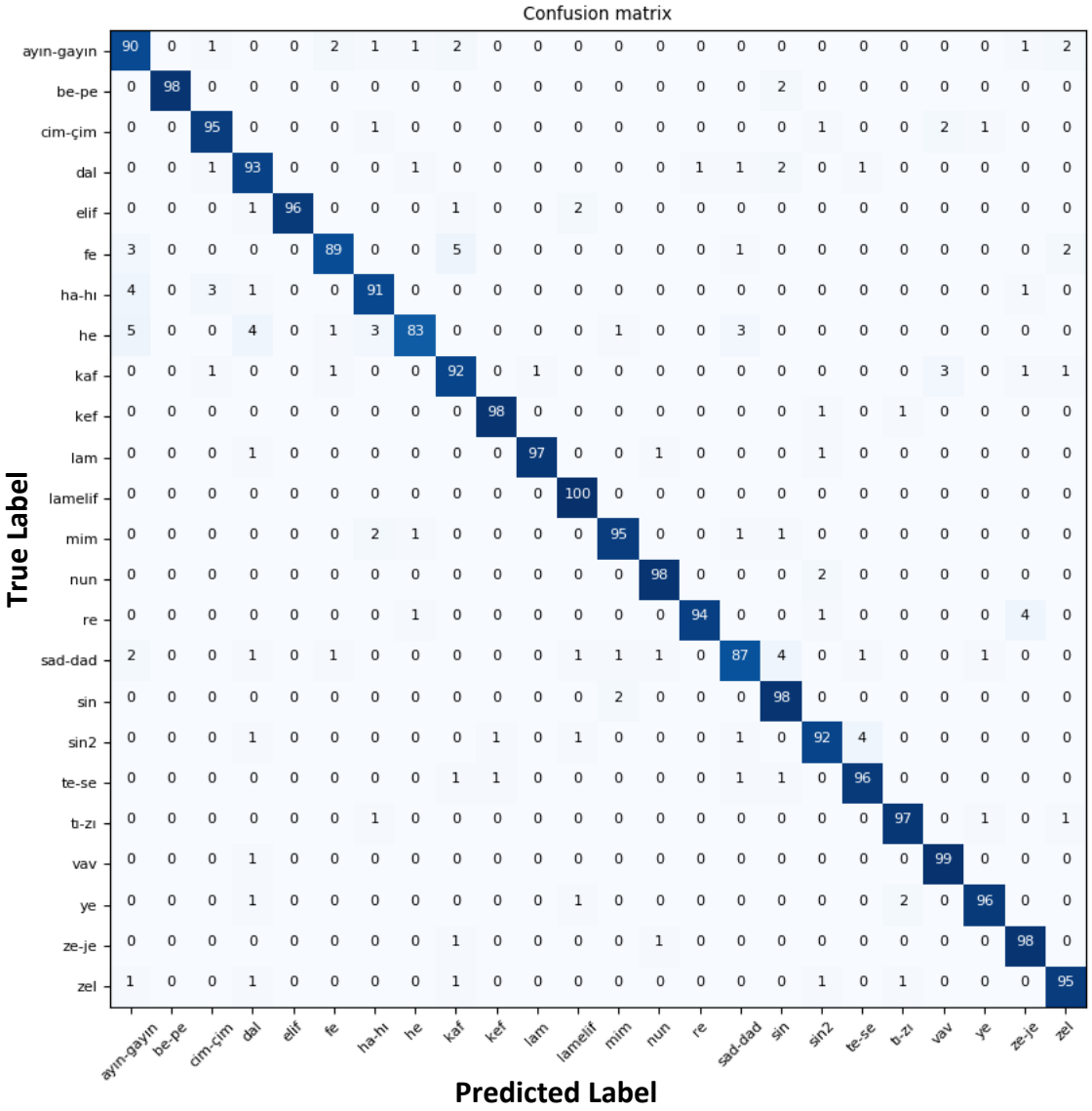



Figure 40. Confusion Matrix of the CNN Model-I (Dataset Separated by 75% to 25%, Epoch=100)

Three consecutive convolutional and max pooling layers followed the output of the first two layers. Then the output was flattened into an array and passed through a fully connected dense layer with 64 hidden units and was connected with 60% dropout layer. Dropout layer was used to prevent overfitting by randomly setting some of the input units to 0 at each update during training. Same step was repeated again with a dense layer and 30% dropout layer. Finally, it was connected with a fully connected dense layer with 24 nodes, which was also the output layer for the model. Figure 41 shows the layers of the CNN Model-II.

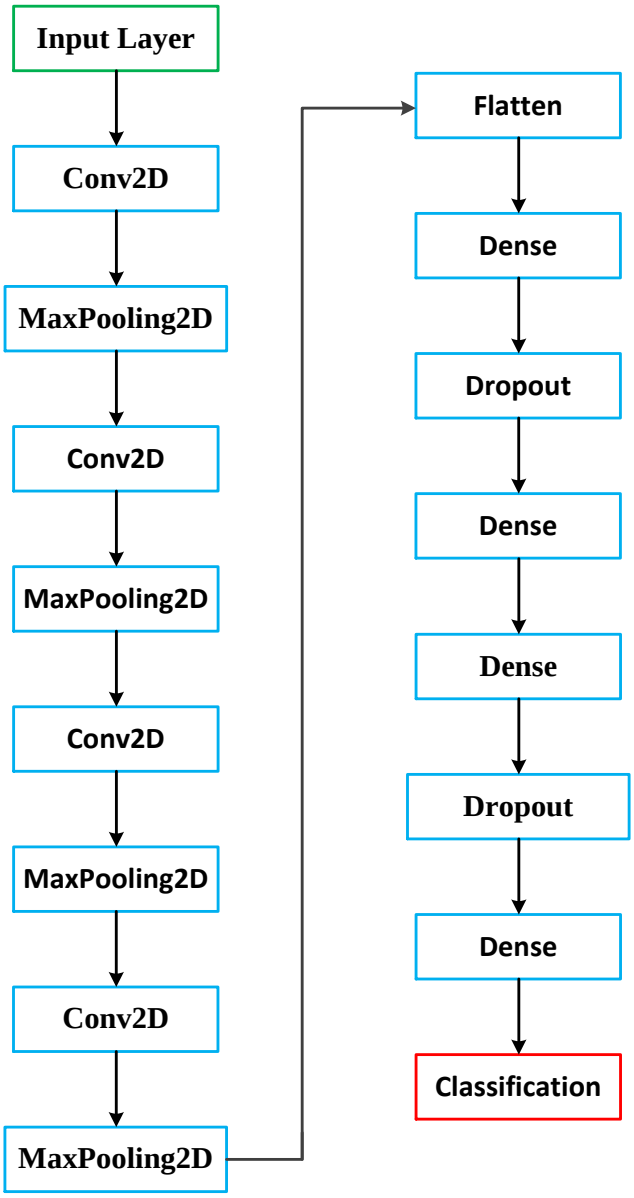



Figure 41. Layers of the CNN Model-II

This model was also trained with the same dataset while it was split by 80% to 20%. As can be seen in the Figure 42 and 43, the accuracy was increased to 95.94% and loss showed significant drop to 16.02%.

Lastly, training was repeated by using 75% of the dataset as training set and 25% as test set. As can be seen from the following Figure 44 and 45, CNN Model-II showed better results for both accuracy and loss. Accuracy of 97.58% was obtained and loss dropped to 9.54%.

Table 5. Architecture of the CNN Model-II

| Layer No (Type) | Output Shape | Parameter |
|---|---|---|
| Conv2D | 150, 150, 32 | 320 |
| MaxPooling2D | 75, 75, 32 | 0 |
| Conv2D | 75, 75, 32 | 9248 |
| MaxPooling2D | 37, 37, 32 | 0 |
| Conv2D | 37, 37, 64 | 18496 |
| MaxPooling2D | 18, 18, 64 | 0 |
| Conv2D | 18, 18, 64 | 36928 |
| MaxPooling2D | 9, 9, 64 | 0 |
| Flatten | 5184 | 0 |
| Dense | 64 | 331840 |
| Dropout | 64 | 0 |
| Dense | 64 | 4160 |
| Dense | 64 | 4160 |
| Dropout | 64 | 0 |
| Dense | 24 | 1560 |
| Total Parameters: 406712<br>Trainable Parameters: 406712<br>Non-trainable Parameters: 0 | | |

Figure 46 and Figure 47 show confusion matrices of the CNN Model-II. Figure 47 shows the confusion matrix where the model showed the best recognition performance. It can be seen from the figure, most of the diagonal cells are very close to one hundred that is the number of each character in the test set.

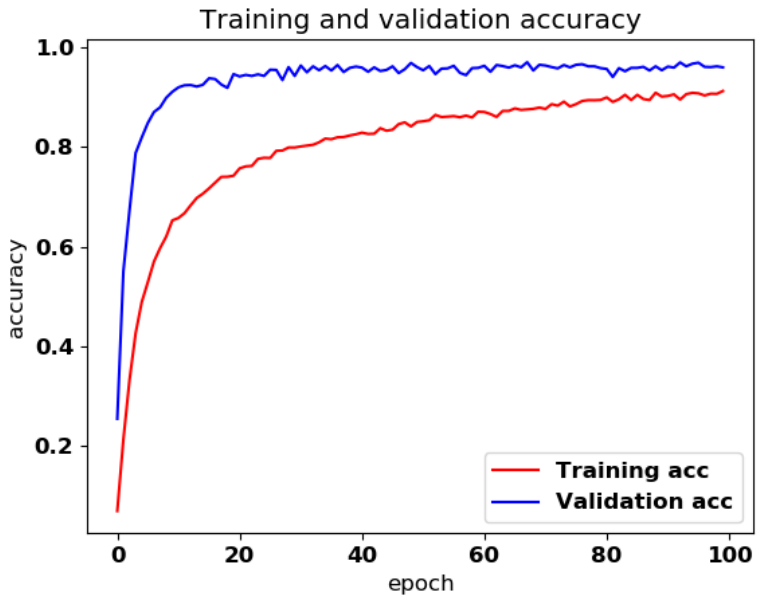



Figure 42. Accuracy Results of the CNN Model-II (Dataset Separated by 80% to 20%)

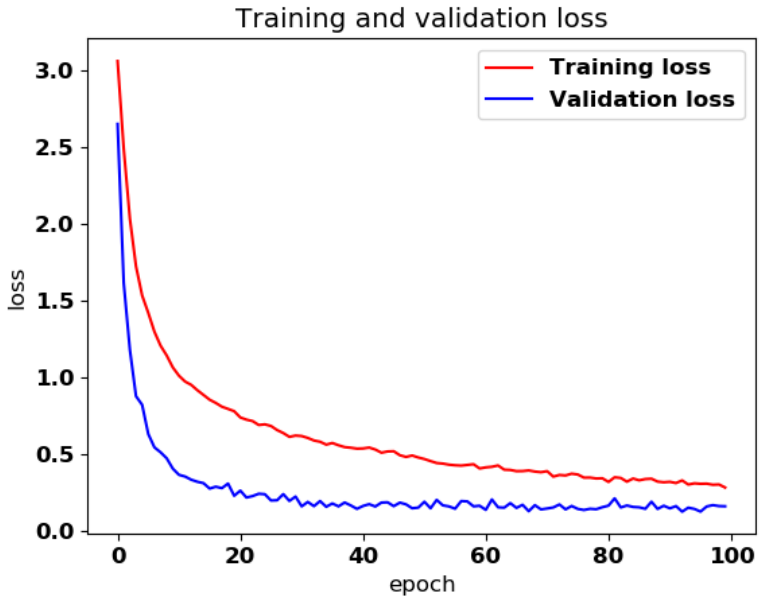



Figure 43. Loss Results of the CNN Model-II (Dataset Separated by 80% to 20%)

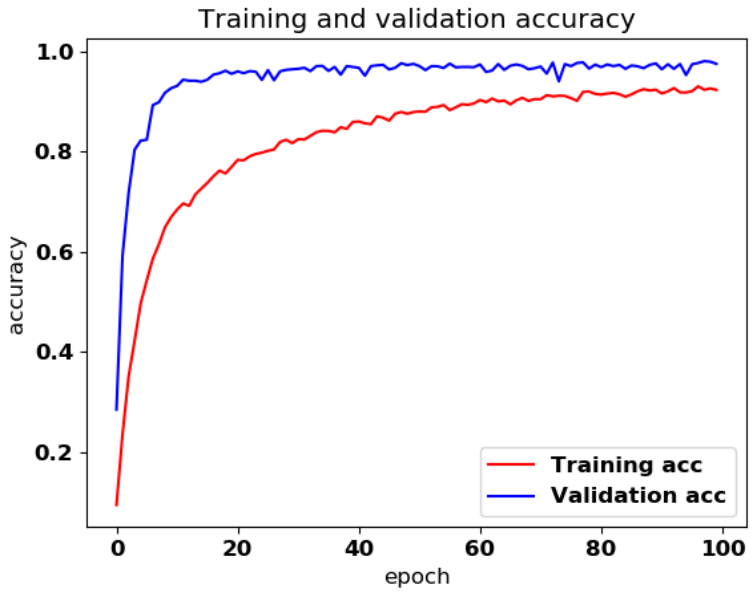



Figure 44. Accuracy Results of the CNN Model-II (Dataset Separated by 75% to 25%)

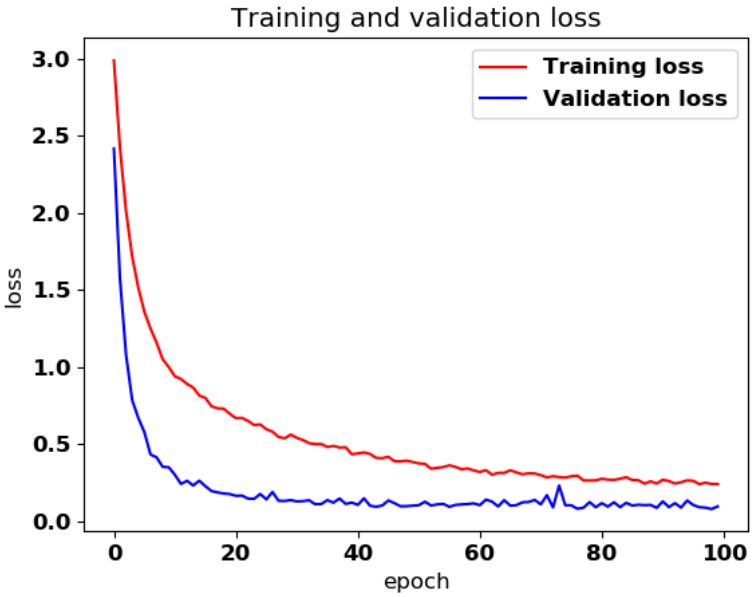



Figure 45. Loss Results of the CNN Model-II (Dataset Separated by 75% to 25%)

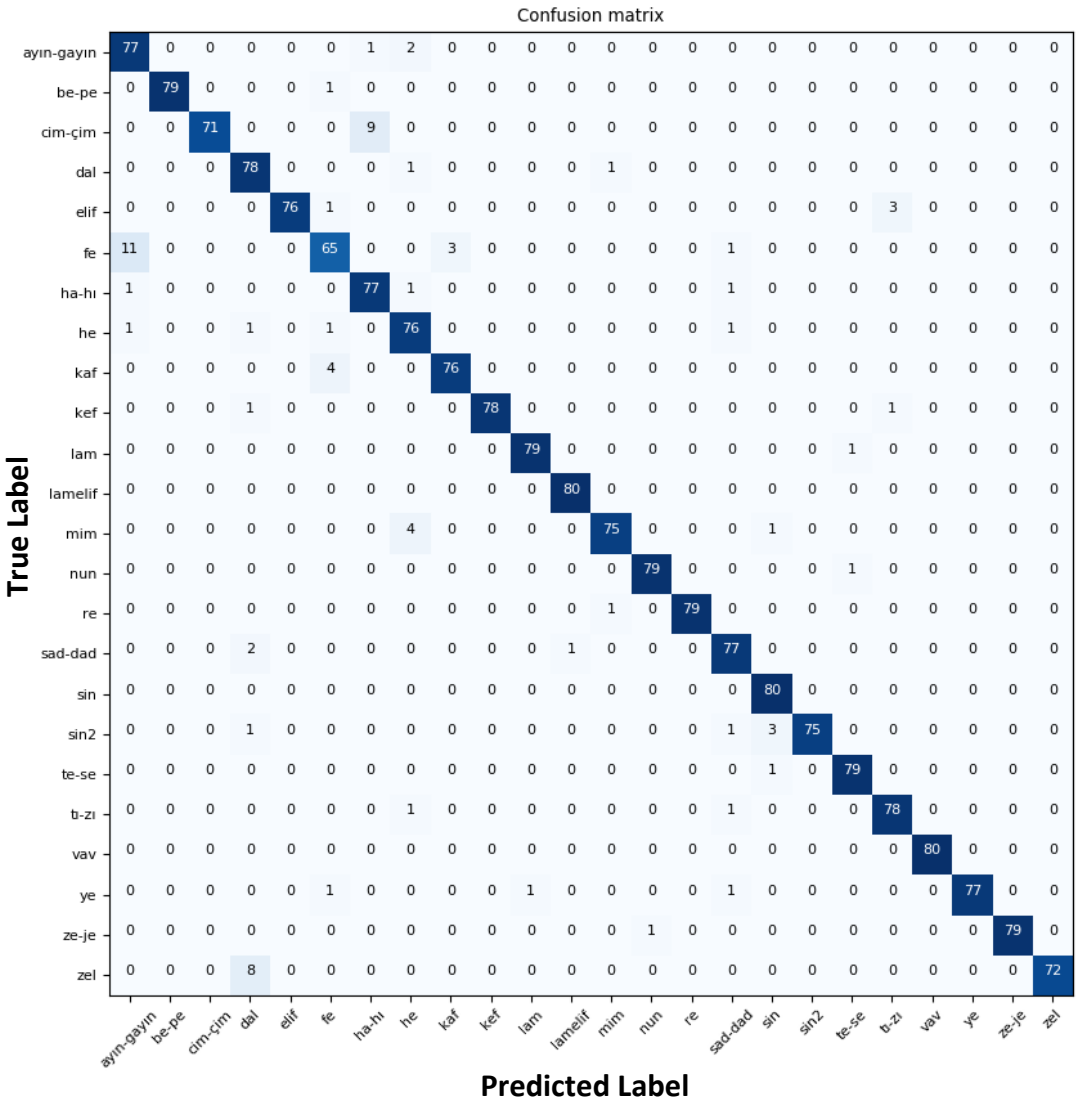



Figure 46. Confusion Matrix of the CNN Model-II (Dataset Separated by 80% to 20%)

### 4.2.3. Comparing CNN Models to Conventional OCR Methods

In order to interpret the results of proposed CNN models, same dataset was tested with the methods that have proven themselves in the OCR field. For this purpose, two different feature descriptors were selected.

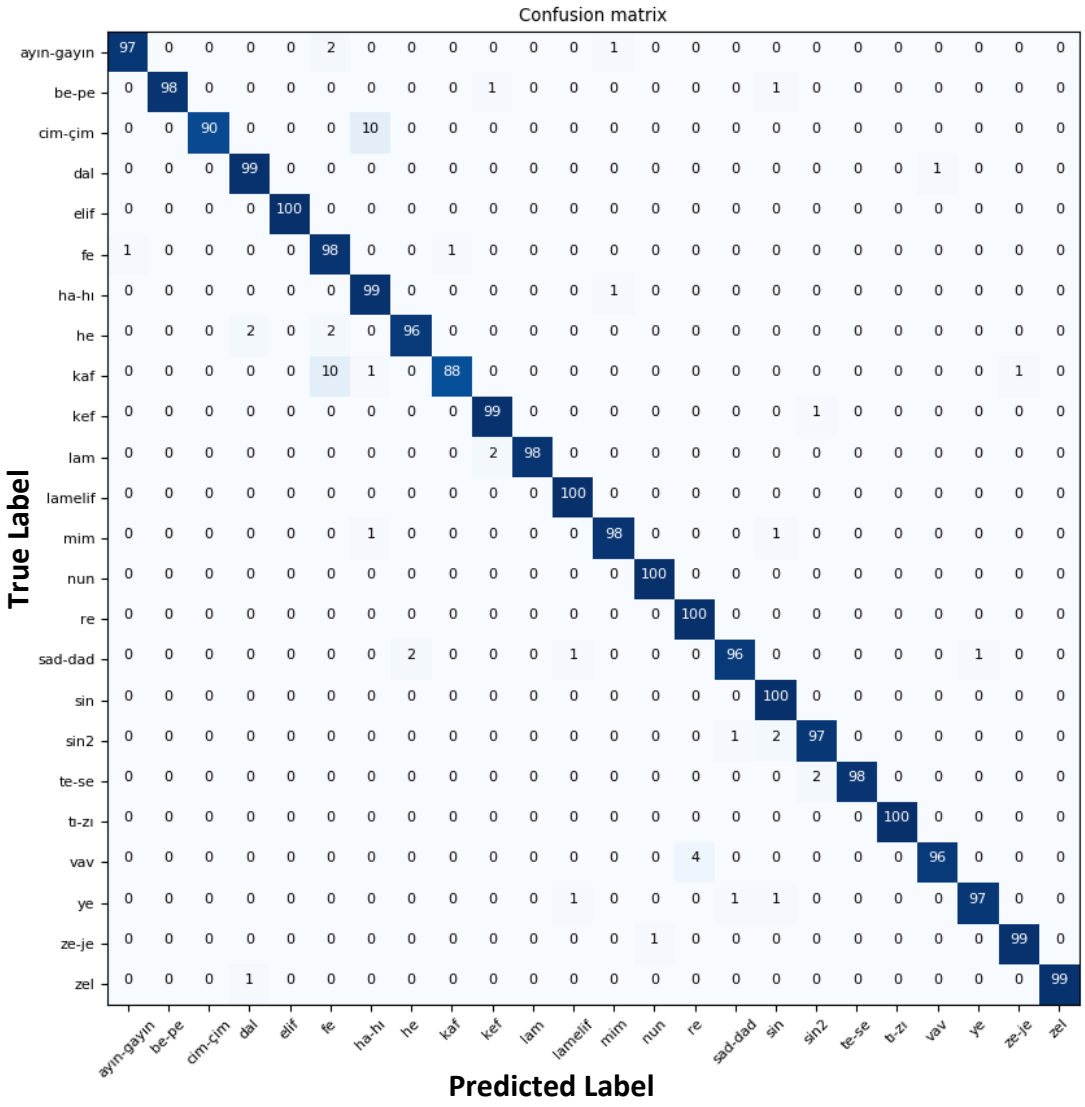



Figure 47. Confusion Matrix of the CNN Model-II (Dataset Separated by 75% to 25%)

## Histogram of Oriented Gradients

The distributions of gradient directions are used as features in the histogram of oriented gradients (HOG). It is implemented with a block which is similar to a sliding window that can be considered as a pixel grid. Horizontal and vertical gradients are calculated for each pixel within this pixel grid. After gradients are constituted, gradient magnitude and gradient angle that form the gradient vectors are calculated and they are compressed into 9 directions that are separated by 20°.

Figures from 48 to 51 show HOG distributions of some letters that belong to Ottoman character dataset created in this study when block size was chosen as 2x2, 4x4 and 8x8 pixels. When the chosen pixel grid sweeps the entire image, HOG features are extracted. Thus, character structure is retained while eliminating insignificant information.

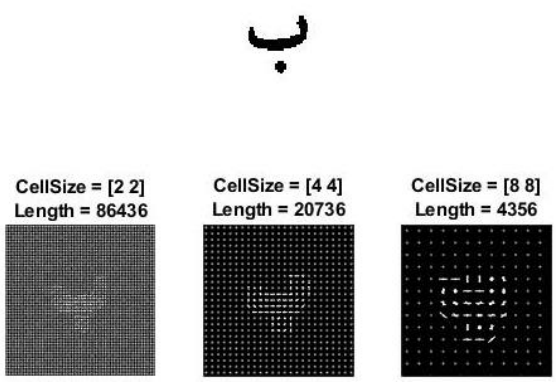



Figure 48. Histogram of Oriented Gradients of Letter “b”

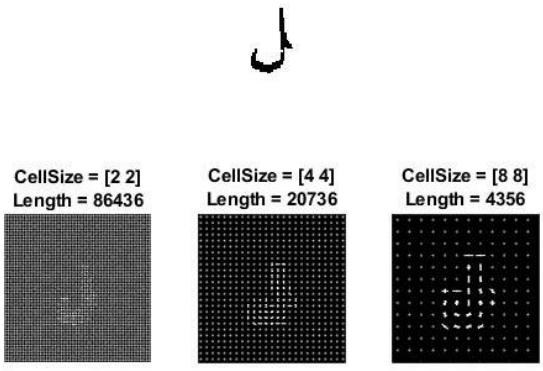



Figure 49. Histogram of Oriented Gradients of Letter “l”

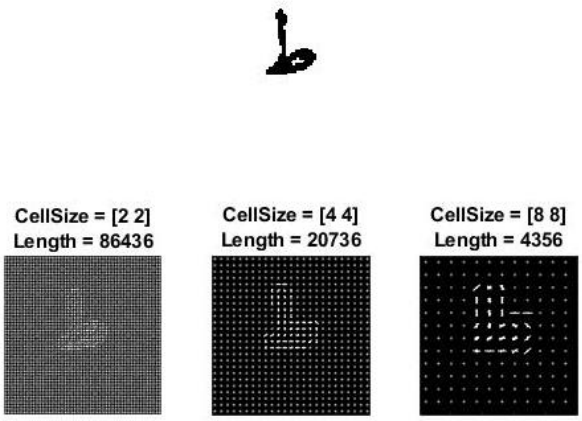



Figure 50. Histogram of Oriented Gradients of Letter “t”

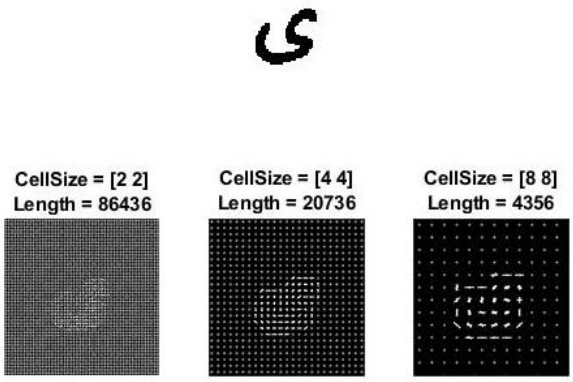



Figure 51. Histogram of Oriented Gradients of Letter “y”

**Principal Component Analysis**

Principal component analysis (PCA) is one of the most commonly used feature extraction technique and it is based on projection of the data to a lower dimensional space in order the retain the information while eliminating insignificant information (Bishop, 2006). PCA is achieved by transforming the data to uncorrelated components that retain most of the variation exist in the original data. Thus, if amount of variance is high, the information carried by that feature will be high (Jolliffe, 2002).

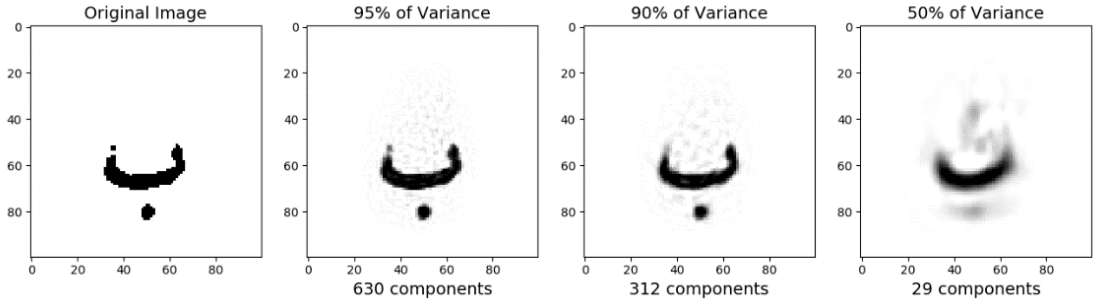



Figure 52. PCA of Letter "b" when the variance is 95%, 90% and 50%

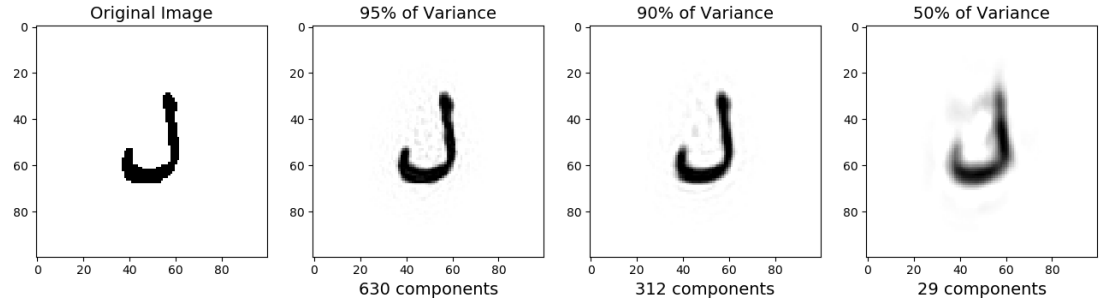



Figure 53. PCA of Letter "l" when the variance is 95%, 90% and 50%

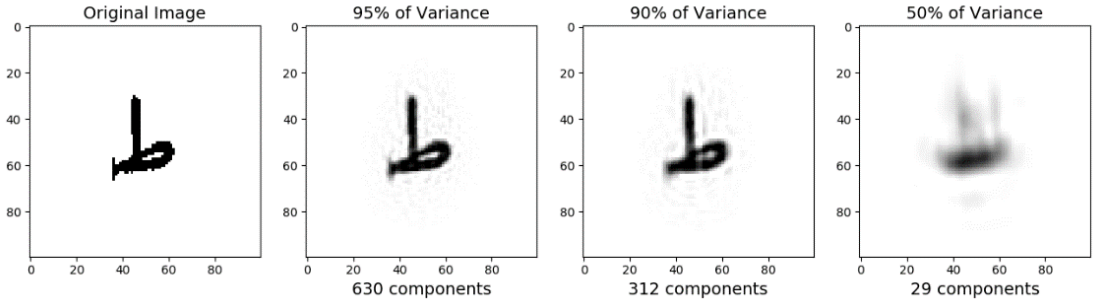



Figure 54. PCA of Letter "t" when the variance is 95%, 90% and 50%

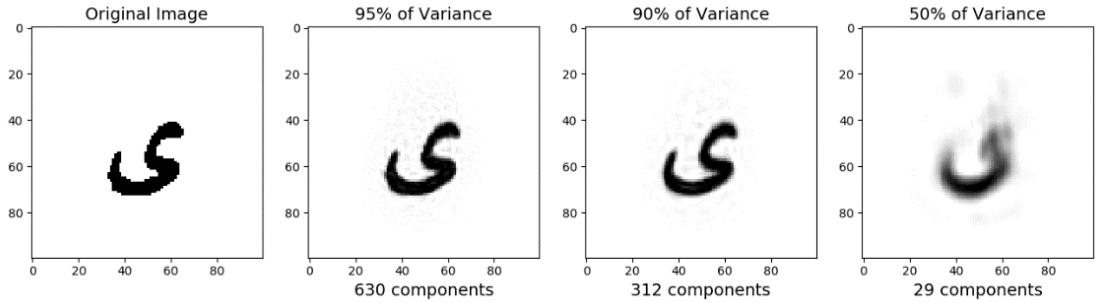



Figure 55. PCA of Letter "y" when the variance is 95%, 90% and 50%

Figures from 52 to 55 show plots of the same characters as in Figure 48 to 51 when the variance is 95%, 90% and 50% respectively. Results showed that 95% of the information is contained in 630 dimensions, 90% of the information is contained in 312 components and 50% of the information is contained in 29 dimensions.

After features were extracted, Ottoman character images were mapped into 24 classes by using two different classifiers called as Support Vector Machines and k-Nearest Neighbors.

## Support Vector Machines (SVM)

The objective of the support vector machine algorithm is to find a hyperplane that separates the data points of classes. Performance of the SVM classifier depends on choosing the correct hyperplane that maximizes the margin between classes. Figure 56 illustrates the optimal hyperplane that separates two classes from the maximum distance that is possible (Duda, Hart, and Stork, 2000).

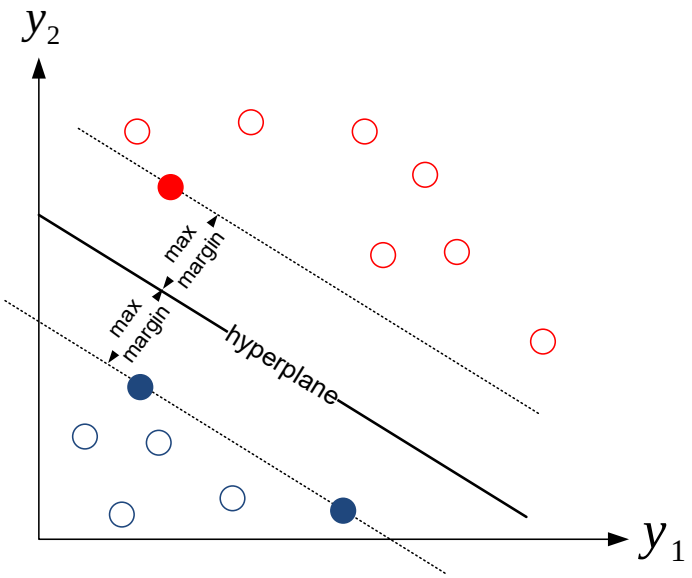



Figure 56. Selection of Hyperplane Between Two Classes in SVM
(Source: Duda, Hart, and Stork, 2000)

## k-Nearest Neighbors (k-NN)

In k-NN algorithm, it is assumed that a data is similar to the data that is in the close proximity. It classifies test data by measuring its distance to each training data by using distance functions such as Euclidian, Manhattan, Minkowski etc. Euclidian distance is the most commonly used distance function and its mathematical model is given below.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

In this study, k is chosen as 3 and Euclidian distance is chosen as the distance function.

Table 6 and Table 7 show the overall recognition results for all methods applied in this study. SVM method showed good performance when histogram of oriented gradients were used to extract features. However CNN Model-II has the best score among them with the accuracy of 97.58%.

Table 6. Recognition Results of the k-NN and SVM Classifiers

| Feature Detector | k-NN Classifier | SVM Classifier |
|---|---|---|
| HOG [4x4] | 93.26% | 96.13% |
| PCA | 91.12% (90% of variance) | 86.62% (95% of variance) |

Table 7. Recognition Results of the CNN Model-I and CNN Model-II

| | 80%-20% | 75%-25% |
|---|---|---|
| CNN Model-I | 84.84% (25 epochs) | 85.25% (25 epochs)<br>94.46% (100 epochs) |
| CNN Model-II | 95.94% (100 epochs) | 97.58% (100 epochs) |

# CHAPTER 5

# CONCLUSIONS AND FUTURE WORK

In this study, deep neural networks were used to recognize printed Ottoman letters. In order to recognize the letters, the deep neural network was trained with character dataset obtained from the segmentation of texts of different lengths selected from various Ottoman documents collected from the Turkish National Library. While creating this dataset, horizontal and vertical projections were used and thus, isolated characters were fragmented. However, a new method has been developed for separating the combined letters due to the adjacent structure of the Ottoman texts. Segmented characters were divided into 24 classes and a total of 9600 images were obtained with 400 images for each class.

After the dataset is obtained, it was divided by 75% to 25% as training and test sets. Then, a CNN model which only has a single convolution layer was built to recognize 24 classes. The network was trained with 7200 images and tested with 2400 images. The same process was repeated by dividing dataset with 80-20%. The results showed that CNN performed better for both training and test sets when dataset is divided by 80% to 20%.

A second multilayer CNN was then created to observe the effect of system depth on recognition performance. This network was also trained by separating the dataset by 75% -25% first and the procedure was repeated for 80% -20% again. It has been observed that the best recognition result of the system was obtained when dataset is divided by 80% to 20% and the network is multi layered.

This study focused only on printed Ottoman documents however it can be expanded by creating new dataset for handwriting samples and proposed CNN models can be evaluated for handwritten Ottoman scripts.

Deep learning models can also be utilized for the recognition of the non-writing objects in the documents in order to fully automate translation of the scanned Ottoman documents.

Finally, recognition of Ottoman characters can be combined with grammar knowledge in order to build an automated system that translates Ottoman text image to the contemporary Turkish.

# REFERENCES


"Harfler" 2019. 2019. https://osmanlicaogren.com/harfler#1561411751942-cc8c8071-aa9f.

Adıgüzel, Hande, Emre Şahin, and Pınar Duygulu. 2012. "A Hybrid Approach for Line Segmentation in Handwritten Documents." In *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, 503–8. https://doi.org/10.1109/ICFHR.2012.156.

Ahmed, Saad Bin, Saeeda Naz, Muhammad Imran Razzak, and Rubiyah Yousaf. 2017. "Deep Learning Based Isolated Arabic Scene Character Recognition." In *IEEE International Workshop on Arabic Script Analysis and Recognition (ASAR)*, 46–51. https://doi.org/10.1109/asar.2017.8067758.

Ali, Asghar, Mark Pickering, and Kamran Shafi. 2018. "Urdu Natural Scene Character Recognition Using Convolutional Neural Networks." In *2nd IEEE International Workshop on Arabic and Derived Script Analysis and Recognition, ASAR 2018*, 29-34. IEEE. https://doi.org/10.1109/ASAR.2018.8480202.

Arifoğlu, Damla, and Pınar Duygulu. 2011. "Word Retrieval In Ottoman Documents." In *19th Signal Processing and Communications Applications Conference*. Antalya. https://doi.org/10.1109/SIU.2011.5929703.

Ashiquzzaman, Akm, and Abdul Kawsar Tushar. 2017. "Handwritten Arabic Numeral Recognition Using Deep Learning Neural Networks." In *IEEE International Conference on Imaging, Vision and Pattern Recognition*, 1–4. IEEE. https://doi.org/10.1109/ICIVPR.2017.7890866.

Ataer, Esra, and Pinar Duygulu. 2006. "Retrieval of Ottoman Documents." *Proceedings of the ACM International Multimedia Conference and Exhibition*, no. January 2006: 155–62. https://doi.org/10.1145/1178677.1178700.

Aydemir, M. Said, Burak Aydın, Hamza Kaya, İbrahim Karlıağa, and Cemil Demir. 2014. “TÜBİTAK Türkçe - Osmanlıca El Yazısı Tanıma Sistemi.” In *22nd Signal Processing and Communications Applications Conference*, 1918–21. IEEE. https://doi.org/10.1109/SIU.2014.6830630.

Baydar, B. 2018. “Convolutional Neural Network Based Brain MRI Segmentation.” Middle East Technical University. http://etd.lib.metu.edu.tr/upload/12622216/index.pdf.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Can, Ethem F., Pinar Duygulu, Fazli Can, and Mehmet Kalpakli. 2010. “Redif Extraction in Handwritten Ottoman Literary Texts.” In *Proceedings - International Conference on Pattern Recognition*, 1941–44. https://doi.org/10.1109/ICPR.2010.478.

“Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences.” 2018. 2018. https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb.

“Converting a Grayscale Image to Binary Image Using Thresholding.” n.d. https://www.geeksforgeeks.org/matlab-converting-a-grayscale-image-to-binary-image-using-thresholding/.

“CUDA GPUs Computing Capabilities.” 2019. 2019. https://developer.nvidia.com/cuda-gpus.

“Dropout Neural Network Layer In Keras Explained.” 2019. 2019. https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab.

Duda, Richard O., Peter E. Hart, and David G. Stork. 2000. *Pattern Classification*. 2nd ed. Wiley.

Elsawy, Ahmed, Hazem M. El-Bakry, and Mohamed Loey. 2017. “Arabic Handwritten Characters Recognition Using Convolutional Neural Network.” *WSEAS Transactions on Computer Research*, no. January 2017.

Haykin, S. 1999. *Neural Networks A Comprehensive Foundation*. 2nd ed. Pearson Education, Inc.

Hsu, Hwei P. (Hwei Piao). 2011. *Schaum’s Outlines: Signals and Systems*.

Islam, Noman, Zeeshan Islam, and Nazia Noor. 2016. “A Survey on Optical Character Recognition System.” *ITB Journal of Information and Communication Technology*, no. December 2016. https://doi.org/10.3850/978-981-09-5346-1_cse-024.

Jain, Anil K., Jianchang Mao, and K. M. Mohiuddin. 1996. “Artificial Neural Networks: A Tutorial.” *Computer* 29 (3): 31–44. https://doi.org/10.1109/2.485891.

Jolliffe, I.T. 2002. *Principal Component Analysis*. 2nd ed. Springer.

Karpathy, A. 2016. “Convolutional Neural Networks for Visual Recognition.” Stanford University CS231n Lecture Notes. 2016. http://cs231n.github.io/.

“Keras: The Python Deep Learning Library.” 2019. 2019. https://keras.io/.

Kingma, Diederik P., and Jimmy Ba. 2014. “Adam: A Method for Stochastic Optimization.” *ICLR*, 1–15. http://arxiv.org/abs/1412.6980.

Kılıç, Niyazi, Pelin Görgel, Osman N. Uçan, and Ahmet Kala. 2008. “Multifont Ottoman Character Recognition Using Support Vector Machine.” In *3rd International Symposium on Communications, Control and Signal Processing*, 328–33. St Julians, Malta: IEEE. https://doi.org/10.1109/ISCCSP.2008.4537244.

Kuo, C. C.Jay. 2016. “Understanding Convolutional Neural Networks with a Mathematical Model.” *Journal of Visual Communication and Image Representation* 41: 406–13. https://doi.org/10.1016/j.jvcir.2016.11.003.

Kurt, Zeyneb, H. Irem Turkmen, and M. Elif Karsligil. 2007. “Ottoman Alphabet Character Recognition by LDA.” *IEEE 15th Signal Processing and Communications Applications*, 1–4. https://doi.org/10.1109/SIU.2007.4298685.

Mcculloch, W., and W. Pitts. 1990. “A Logical Calculus of the Ideas Immanent in Nerous Activity.” *Bulletin of Mathematical Biology* 52 (1/2): 99–115.

Nalbant, Muammer, Mustafa Burunkaya, and Yılmaz Eroğlu. 2009. “Osmanlica Elyazisi Harfleri Çevrimiçi Tanima.” *E-Journal of New World Sciences Academy* 4 (2): 148-64.

Onat, Ayşe, Ferruh Yıldız, and Mesut Gündüz. 2008. “Ottoman Script Recognition Using Hidden Markov Model.” *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering* 2 (2): 462–64. https://doi.org/10.5281/zenodo.1059657.

“Ottoman Turkish.” 2010. 2010. https://www.loc.gov/catdir/cpso/romanization/ottoman.doc.

Öztürk, Ali, Salih Güneş, and Yüksel Özbay. 2000. “Multifont Ottoman Character Recognition.” *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems* 2: 945–49. https://doi.org/10.1109/ICECS.2000.913032.

Şahin, Emre, Hande Adıgüzel, Pınar Duygulu, and Mehmet Kalpaklı. 2012. “OTAP Osmanlıca Metinleri Internet Arayüzü.” In *20th Signal Processing and Communications Applications Conference*. https://doi.org/10.1109/SIU.2012.6204792.

Sarfraz, M., S. N. Nawaz, and A. Al-Khuraidly. 2003. “Offline Arabic Text Recognition System.” *Proceedings - 2003 International Conference on Geometric Modeling and Graphics, GMAG 2003*, 30–35. https://doi.org/10.1109/GMAG.2003.1219662.

Şaykol, Ediz, Ali Kemal Sinop, Uğur Güdükbay, Özgür Ulusoy, and A. Enis Çetin. 2004. “Content-Based Retrieval of Historical Ottoman Documents Stored as Textual Images.” *IEEE Transactions on Image Processing* 13 (3): 314–25. https://doi.org/10.1109/TIP.2003.821114.

“Sırat-ı Müstakim.” 1908. *Sırat-ı Müstakim*.

Stenroos, Olavi. 2017. “Object Detection from Images Using Convolutional Neural Networks.” Aalto University.

“The Ultimate Guide to Convolutional Neural Networks (CNN).” 2018. 2018. https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn.

“Training Deep Neural Networks.” 2018. 2018. https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964.

Tulum, Mertol. 2009. *Osmanlı Türkçesine Giriş*. Edited by Albdülkadir Gürer. 4. Baskı. T.C. Anadolu Üniversitesi.

“Using Neural Nets to Recognize Handwritten Digits.” 2019. 2019. http://neuralnetworksanddeeplearning.com/chap1.html.

Wu, J. 2017. “Introduction to Convolutional Neural Networks.” *National Key Lab for Novel Software Technology*, 1–31. https://doi.org/10.1007/978-3-642-28661-2-5.

Yalnız, İsmet Zeki, İsmail Şengör Altıngövde, Uğur Güdükbay, and Özgür Ulusoy. 2009. “Ottoman Archives Explorer: A Retrieval System for Digital Ottoman Archives.” *Journal on Computing and Cultural Heritage* 2 (3). https://doi.org/10.1145/1658346.1658348.

Younis, Khaled, and Abdullah Khateeb. 2017. “Arabic Hand-Written Character Recognition Based on Deep Convolutional Neural Networks.” *Jordanian Journal of Computers and Information Technology* 3 (3): 186. https://doi.org/10.5455/jjcit.71-1498142206.