KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

HANDWRITTEN TEXT RECOGNITION USING DEEP LEARNING

MASTERS OF SCIENCE THESIS

Usama MUNIR

JULY 2019
TRABZON

KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPRATMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

HANDWRITTEN TEXT RECOGNITION USING DEEP LEARNING

Usama MUNIR

This thesis is accepted to give the degree of
"MASTER OF SCIENCE"
By
The Graduate School of Natural and Applied Sciences at
Karadeniz Technical University

The Date of Submission	: 21 / 05 /2019
The Date of Examination	: 21 / 06 /2019

Supervisor : Assistant Prof. Dr. Mehmet ÖZTÜRK

Trabzon 2019

KARADENİZ TECHNICAL UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

Department of Electrical & Electronics Engineering
Usama MUNIR

HANDWRITTEN TEXT RECOGNITION USING DEEP LEARNING

Has been accepted as a thesis of

MASTER OF SCIENCE

after the Examination by the Jury Assigned by the Administrative Board of the Graduate School of Natural and Applied Sciences with the Decision Number 1806 dated 28 / 05 / 2019

Approved By

Chairman	:	Associate Prof. Dr. Nur Hüseyin KAPLAN	[illegible]
Member	:	Associate Prof. Dr. Önder AYDEMİR	[illegible]
Member	:	Assistant Prof. Dr. Mehmet ÖZTÜRK	[illegible]

Prof. Dr. Asim KADIOĞLU
Director of Graduate School

PREFACE

First and foremost, I am very grateful to Allah, the Almighty, the guide of all guides, for giving me the strength and energy to undergo all the challenges in life and for blessing me with the courage to complete this work.

I would like to express my deepest and profound respect to my supervisor, Assistant Prof. Dr. Mehmet Öztürk for his supervision, guidance and support from day one till the end. He stood beside me in every difficult situation and helped me a lot during my research work.

I would like to thanks my family, especially my parents whose prayers and hard work made it possible to complete my masters today.

I would like to thank Mr. Shifat Kayser, a student at Karadeniz Technical University, for allowing me to use scanned versions of his class notes for my research work.

Last but not the least I would like to thank the Turkish Government, Turkey Scholarship and Karadeniz Technical University to provide me an opportunity to do my masters from here.

Usama MUNIR

Trabzon 2019

DECLARATION

I, the undersigned, declare that this study entitled “Handwritten Text Recognition Using Deep Learning”, completed under the supervision of Assistant Prof. Dr. Mehmet ÖZTÜRK, is my original work and has not been presented for a degree in any other university and that all sources of materials used for the study have been duly acknowledged. 21/06/2019

Usama MUNIR

TABLE OF CONTENTS

Master Thesis

SUMMARY

HANDWRITTEN TEXT RECOGNITION USING DEEP LEARNING

Usama MUNIR

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Department of Electrical & Electronics Engineering
Supervisor: Assistant Prof. Dr. Mehmet ÖZTÜRK
2019, 65 Pages

Handwritten alphabet recognition is one of the Artificial Intelligence applications which provides important fundamental for various advanced applications, including information retrieval and human-computer interaction applications. This thesis seeks to classify an individual handwritten character so that handwritten text can be translated to a digital form.

To classify a complete word or text, the first and foremost step is the accurate detection of text lines. A text line detection system is developed which can detect all text lines based on the skew angle of text lines by dividing the original image of an $A4$ size scanned document. Individual alphabets are detected from each text line at a later stage to give the input to a deep neural network for recognition.

A dataset of our own handwriting is also prepared that includes 2200 images of each alphabet, which is mixed with another publicly available dataset for the training phase of the deep learning network. A total of $26 \times 7800 = 202,800$ images are used for the training of the neural network.

A GUI system using MATLAB is developed which can input a scanned document in image form, and can give an output of text lines detection, alphabet detection and alphabets recognition. A set of different parameters can also be changed to get the desired output depending upon the variations in different types of documents.

Key Words: Artificial Intelligence, Deep Learning, Machine Learning, Handwritten character recognition, MATLAB.

Yüksek Lisans Tezi

ÖZET

HANDWRITTEN TEXT RECOGNITION USING DEEP LEARNING

Usama MUNIR

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği
Danışman: Dr. Öğr. Üyesi Mehmet ÖZTÜRK
2019, 65 Sayfa

El yazısı karakter tanıma, bilgi çıkarma ve insan-bilgisayar etkileşimi uygulamaları da dahil olmak üzere çeşitli gelişmiş uygulamalar için önemli bir temel sağlayan yapay zeka uygulamalarından biridir. Bu tez, el yazısı metninin dijital bir forma çevrilebilmesi için tek bir el yazısı karakterini sınıflandırmaya çalışır.

Tam bir kelimeyi veya metni sınıflandırmak için, öncelikle yapılması gereken adım metin satırlarının doğru algılanmasıdır. $A4$ boyutunda taranmış bir belgenin orijinal görüntüsünü bölerek metin satırlarının eğrilik açısına göre tüm metin satırlarını algılayabilen bir metin satırı algılama sistemi gel- iştirilmiştir. Metin satırlarında yer alan her bir harf görüntüsü tespit edilerek daha sonra tanınması için derin öğrenme ağına giriş olarak verilir.

Derin öğrenme ağının eğitim aşaması için kamuya açık bir veri setiyle birlikte kullanılan ve her bir harfin 2200 görüntüsünü içeren kendi el yazımızdan oluşturulmuş bir veri kümesi de hazırlanmıştır. Yapay sinir ağının eğitimi için toplamda $26 \times 7800 = 202,800$ adet resim kullanılmıştır.

MATLAB kullanılarak taranan bir dokümanı görüntü biçiminde alabilen ve metin satırları algılaması, harf tespiti ve harf tanıma sonuçlarını verebilen bir GUI sistemi tasarlanmıştır. Çeşitli belge türlerindeki farklılıklara ve içerdikleri gürültüye bağlı olarak istenilen sonucu elde etmek için farklı parametre de ayarlanabilmektedir.

Anahtır Kelimeler: Yapay zeka, Derin öğrenme, Derin öğrenme, Makine öğrenme, El yazısı karakter tanıma, MATLAB.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
TL	Transfer Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
GUI	Graphical User Interface
OCR	Optical Character Recognition
IAM	Informative Applied Mathematics
CMATER	Center for Microprocessor Application for Training Education and Research
IC	Integrated Circuits
NIST	National Institute of Standards and Technology
MNIST	Modified National Institute of Standards and Technology
EMNIST	Extended Modified National Institute of Standards and Technology
SWT	Stationary Wavelet Transform
VLSI	Very Large Scale Integration
SOM	Self Organizing Map
RGB	Reg Green Blue
GPU	Graphics Processing Unit
CPU	Central Processing Unit

1. INTRODUCTION AND LITERATURE REVIEW

Communication has been in use since the birth of first mankind on earth. In the start, people had some sign language for intercommunications which is still valid for deaf people living across the world. People from different regions have different communication languages and different writing styles. Some languages have a systematic way of writing while others have some random or graphical way of writing.

The invention of very first Egyptian and Sumerian scripts of writing dates back to some 3400 to 3200 BC during the Egyptian era. Chinese script was invented around 1200 BC and is believed to be one of the first independent scripts of that era [1]. The first true alphabetical script and is still in use dates back to 800 BC, belongs to the Greek language [2]. But the most commonly used writing style which evolved around 300 BC is Latin, which is widely used around the world by more than 70 % of the world population [3].

Scientists, philosophers, teachers and other professionals had been writing their work by hand until the invention of the typewriter in 1886 [4], which was a breakthrough for writers to write on a big scale. The typewriter quickly became an indispensable tool for writing all official documents. It was widely used by professional writers, in offices, and for business correspondence in private homes [5].

The typewriter had been in big use in the 20^{th} century until the invention of modern computers and Integrated Circuits (ICs) in 1970. As time passed, computers limited the use of typewriters from offices and homes. Because, computers were more easy, flexible, cost-saving and a multipurpose machine to be used in offices. Later on, as upgradation of computers continued, it became cheaper and as a result, it became an essential part of every home and office.

Despite our life is surrounded by technologies in today' s world, thus, writing on portable computers or smart phones is very easy, still, on a lot of occasions, every human have to write a lot of things by hand, whether it is in the form of lecture notes, meeting notes or writing contact details etc. As it is human nature to upgrade everything they have, it also became important to make some program which can digitize handwritten text.

Early Optical Character Recognition (OCR) may be traced back to the technologies involving telegraphy and creating reading devices for blinds [6]. Commercial products incorporating handwriting recognition as a replacement for keyboard input were introduced in the early 1980s.

1.1. Artificial Intelligence

Artificial intelligence (AI) also referred to as Machine Intelligence sometimes is an area of computer science that focuses on the creation of intelligence, where machines try to behave like humans. Any technology that includes some sort of intelligence can be referred to as Artificial Intelligence.

Computer science defines AI research as the study of “intelligent agents”: any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals [7]. More specifically, Kaplan and Haenlein [8] define AI as “a system’s ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation”.

The classification, shown in Fig. 1.1 may not be absolute as the laws of nature, but it is widely accepted. In general Deep Learning (DL) is a kind of Machine Learning (ML) and Machine Learning is a kind of Artificial Intelligence (AI) [9].

The overall research goal of AI is to create a system that tries to work in an intelligent manner, by the use of previous experience (training). The general problem of simulating (or creating) intelligence has been broken down into sub-problems. These consist of particular traits or capabilities that researchers expect an intelligent system to display. The traits described below have received the most attention [10]:

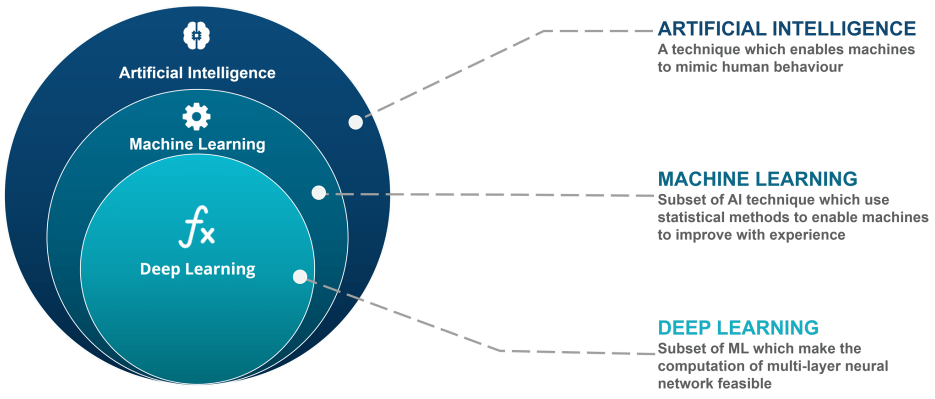

Artificial Intelligence
Machine Learning
Deep Learning
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour
MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience
DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Figure 1.1. Artificial Intelligence, Machine Learning and Deep Learning.

(i) Knowledge

(ii) Reasoning

(iii) Problem-solving

(iv) Perception

(v) Learning

(vi) Planning

(vii) Ability to manipulate and move objects

1.1.1. Machine Learning

The term Machine Learning refers to the automated detection of meaningful patterns in data [11]. Machine learning is an application of Artificial Intelligence that makes a system able to automatically learn from its previous experience (training phase) and thus improve its output accuracy without being explicitly programmed. Machine learning focuses on the development of computer programs that can use the training data, to learn for themselves and give outputs for the next data.

Nowadays Machine Learning is widely used around the globe in various applications, which include search engines where we get our desired results based on our previous searches, our location, placing profitable ads on top based on our current location, adaptive user interfaces, personalized assistants (information systems), anti-spam software learns to filter our email messages between spam and non-spam emails, and credit card transactions (secured by software that learns how to detect frauds) [11].

Machine learning algorithms are often categorized as supervised or unsupervised[11]:

Supervised Learning: Supervised learning is a data mining task of inferring a function from labeled training data. In supervised learning, each example of training data is a pair that consists of an input object and the desired output value called label in ML terminology. Learning algorithm seeks a function or connection between inputs and their respective target outputs (labels). Then this function is applied to the next input whether it can predict its label correctly or not.

Unsupervised Learning: In unsupervised learning the training data given for learning is unlabeled, and the task is to find hidden structure or connection in unlabeled data. Most importantly, unsupervised learning is a type of clustering, which will create different clusters of inputs based on the similarity in the data and will be able to put any new input in an appropriate cluster made earlier during learning. Since the data is unlabeled, there is no error or accuracy to evaluate a potential signal.

1.1.2. Deep Learning

The most important aspect of Deep Learning is its human-like performance. Every task we do and everything we save in our memory is controlled by our nervous system which is composed of neurons [12]. The information processed by our brains is done by the interconnections between different neurons and their relative weights. That is the concept behind the logic of DL. Deep Neural Networks (DNN) are just a group of neurons (like the brain neurons), where the interconnections and their relative strengths are calculated during the training of the networks. Fig. 1.2 shows a basic example of the neural network.

The interesting concept of DNNs is that the networks are independent of the job it has to do. The only thing differs is interconnections and relative strengths between

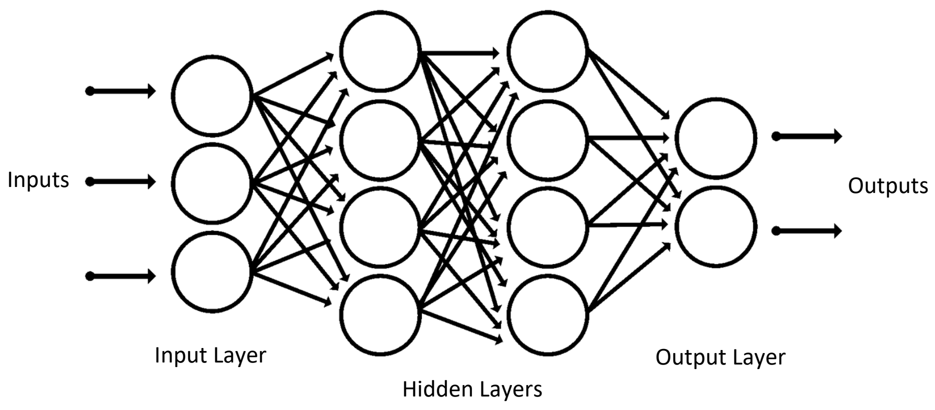


Figure 1.2. A basic machine learning network.

different neurons. So, a neural network developed to do a task may also work on different data or environment, provided it gives good training accuracy on the new data [12].

Deep learning is becoming very useful nowadays due to its better accurate results that were not possible before. Deep Learning is the main technology, which is being used and is still in progress, behind driverless cars. In driverless cars, the deep learning enables them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers.

Considering Fig. 1.2, in DL, the hidden layers can be as many as tens or hundreds. These hidden layers actually make the network more accurate, complex and power consuming.

1.1.3. Machine Learning Vs Deep Learning

Difference between both the terminologies is shown in Fig. 1.3. A Feature is an individual measurable property or characteristic of a phenomenon being observed [13]. In machine learning, features have to be fed to the network by the user, which the computer will try to find in the input data for deciding its output. The network will look for those specific features in the input and decide the relationship between the features and possible outputs. Whereas in deep learning, features are not required to be distinguished by the user, rather the deep learning network will do it. It will look and will try to sort out some features for different types of outputs.

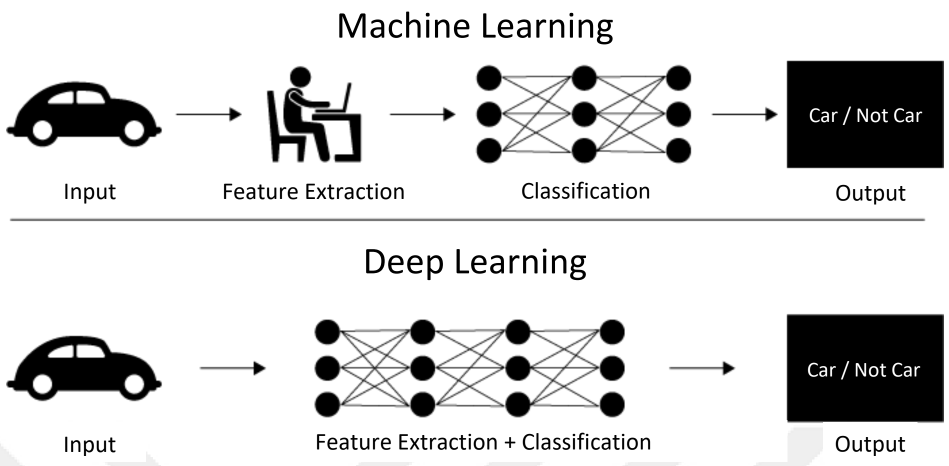


Figure 1.3. Difference between machine learning and deep learning.

1.1.4. Transfer Learning

Transfer Learning (TL), also called ML's next frontier, is a technique which stores solution of one problem and uses it again another similar problem. In ML context, TL is the improvement of learning in a new task with new data, through the transfer of knowledge from a related task that has already been learned and trained [14]. Fig. 1.4 shows a simple difference between machine learning and transfer learning.

Transfer learning is the reuse of a pre-trained model on a new problem. It is currently very popular in the field of DL because it enables us to train DNN with comparatively little data and in less time. This is very useful since most real-world problems typically do not have millions of labeled data to train such complex models [15].

Human learners appear to have inherent ways to transfer knowledge between different tasks. Humans can apply the result of one occurrence on a new incident to improve their own outputs. That is, humans recognize and apply relevant knowledge from other learning experiences when they encounter some new but similar problem. The more related a new task is to the previous task, the easier will it be to tackle.

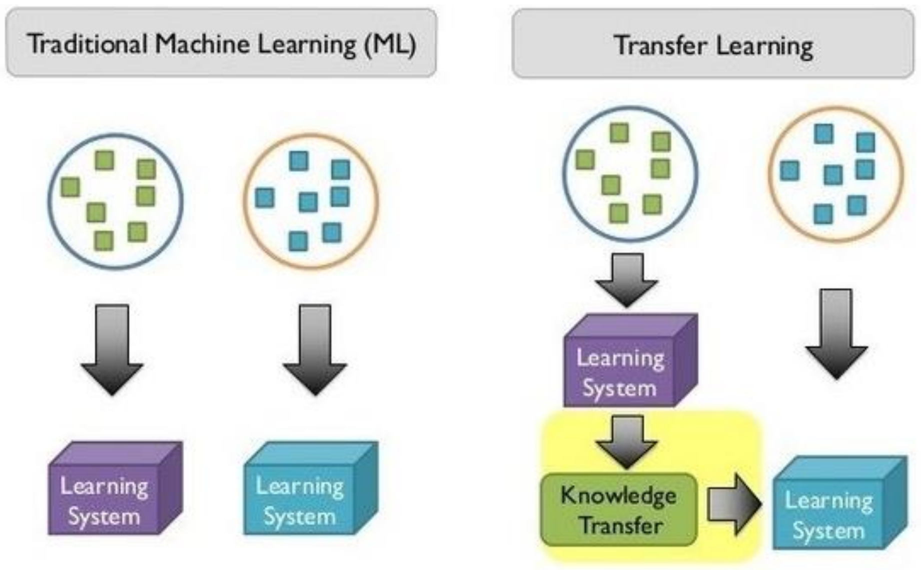


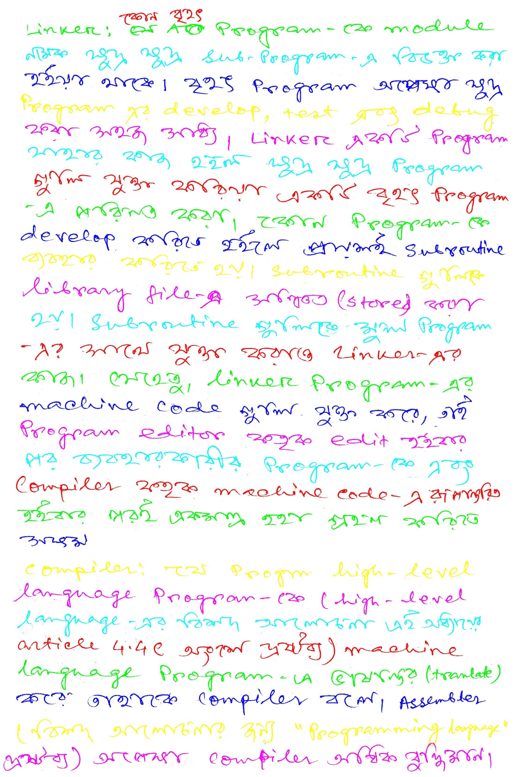
Figure 1.4. Difference between conventional ML and Transfer Learning.

1.2. Handwritten Text Detection

For a text to be recognized from a handwritten document for digitization, the first and foremost task is to detect the text region out of that document accurately. There are some documents that are as simple as Fig. 1.5a (Bangla language [16, 17]) or complex as Fig 1.5b (class notes). These are just two example images, as some text document may be simpler or might the more complex than the one presented here.

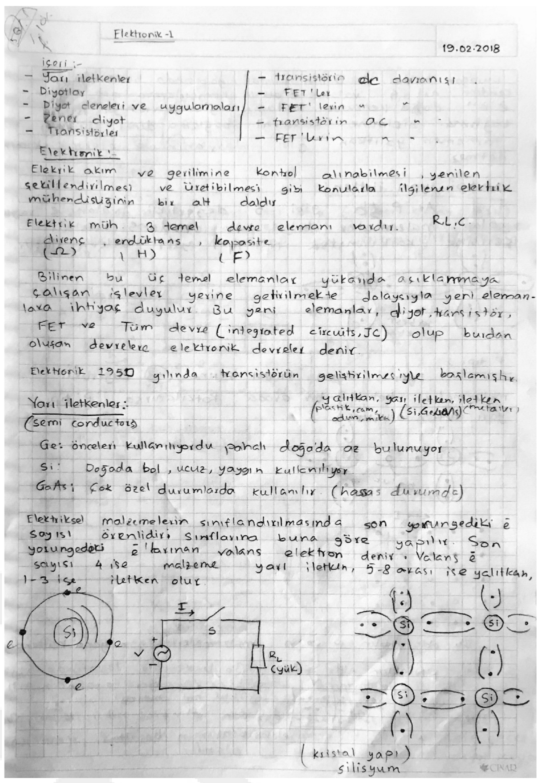
Text region detection may be simple in Fig. 1.5a as there are only text-lines with no figures and plots in it. But still it is a difficult task, as the text lines may not be straight and horizontal, or the alphabets might be overlapping with the text lines nearer to them, or the alphabets might be overlapping with each other.

Text region detection in Fig. 1.5b is a lot more difficult than the previous one. There are some non-text objects within text regions, which makes it more difficult to classify between them. The first step in this case for text region detection is to remove the non-text objects. Rest of the document will be the same as Fig. 1.5a.


Linker: ~~এক এটা~~ কোন বড় Program-কে module নামক ছোট ছোট sub-Program-এ বিভক্ত করা হয়ে থাকে। বড় Program অপেক্ষা ছোট Program এর develop, test এবং debug করা অনেক সহজ। Linker এরূপ Program সমূহের কাজ হলো ছোট ছোট Program গুলোকে যুক্ত করে একটি বড় Program-এ পরিণত করা। কোন Program-কে develop করতে হলে প্রায়শই subroutine ব্যবহার করতে হয়। subroutine গুলোকে library file-এ সংরক্ষিত (stored) থাকে। Subroutine গুলোকে মূল Program-এর সাথে যুক্ত করাও Linker-এর কাজ। মোটকথা, Linker Program-এর machine code গুলো যুক্ত করে, তাই Program editor দ্বারা edit করার পর ব্যবহারকারীর Program-কে প্রথমে Compiler দ্বারা machine code-এ রূপান্তরিত হওয়ার পরই একত্রিত হয়ে গ্রহণ করিতে সক্ষম

Compiler: যে Progrm high-level language Program-কে (high-level language-এর বিস্তারিত আলোচনা এই অধ্যায়ের article 4.4c অংশে দ্রষ্টব্য) machine language Program-এ রূপান্তর (translate) করে তাকে Compiler বলে। Assembler (বিস্তারিত আলোচনার জন্য "Programming language" দ্রষ্টব্য) অপেক্ষা compiler অধিক বুদ্ধিমান।

(a)


Elektronik-1

19.02.2018

içeri :-

- Yarı iletkenler
- Diyotlar
- Diyot deneleri ve uygulamaları
- Zener diyot
- Transistörler
- transistörin dc davranışı
- FET'ler
- FET'lerin " "
- transistörin ac "
- FET'lerin " "

Elektronik :-

Elekrik akım ve gerilimine kontrol alınabilmesi, yenilen şekillendirilmesi ve üretibilmesi gibi konularla ilgilenen elektrik mühendisliğinin bir alt daldır

Elektrik müh. 3 temel devre elemanı vardır. R,L,C.
direnç, endüktans, kapasite
(Ω) (H) (F)

Bilinen bu üç temel elemanlar yukarıda açıklanmaya çalışan işlevler yerine getirilmekte dolaysıyla yeni elemanlara ihtiyaç duyulur. Bu yeni elemanlar, diyot, transistör, FET ve Tüm devre (integrated circuits, JC) olup burdan oluşan devrelere elektronik devreler denir.

Elektronik 1950 yılında transistörün geliştirilmesiyle başlamıştır.

Yarı iletkenler:
(semi conductors)

yalıtkan, yarı iletken, iletken
(plastik, cam, odun, mika) (Si, Ge, GaAs) (metaller)

Ge: önceleri kullanılıyordu pahalı doğa'da az bulunuyor

Si: Doğada bol, ucuz, yaygın kullanılıyor

GaAs: Çok özel durumlarda kullanılır. (hassas durumda)

Elektriksel malzemelerin sınıflandırılmasında son yörüngedeki ē sayısı önemlidir; sınıflarına buna göre yapılır. Son yörüngedeki ē'lerinen valans elektron denir. Valans ē sayısı 4 ise malzeme yarı iletken, 5-8 arası ise yalıtkan, 1-3 ise iletken olur.

(b)

Figure 1.5. Scanned text documents of (a) Bangla language and (b) Class notes.

1.3. Handwritten Text Recognition

In general, handwritten text recognition is classified into two groups which are on-line and off-line handwriting text recognition methods. In off-line recognition, the writing is usually captured optically by a scanner and the complete handwritten document is available as an image in one go, whereas, in on-line recognition system, the two dimensional coordinates of successive points are represented as a function of time and the order of strokes made by the writer are also available. On-line methods are considered to be superior to their off-line counterparts in recognizing handwritten characters due to the fact that, more information is available in on-line recognition systems in form of their writing sequence[18, 19].

1.4. Optical Character Recognition

Optical Character Recognition (OCR) is the recognition or reading of the text from a scanned document and transferring it to digital format, such that the system can read and understand the alphabets written on it. OCR makes it possible, to scan that paper, which someone lost from his/her hard drive, but fortunately placed it in his cupboard in printed form. A scanned document given to a system with no OCR is just an image, without any retrieval of information from it. The computer has no idea that there is some text in that scanned document unless an OCR software is used to convert that image into a texture image, from where the computer can now understand that there is some text actually in that image and this text now can be used by some word processing program to prepare a new document. A more advanced OCR program can even keep the formatting of the document intact while digitizing that text. Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for blinds [6].

1.5. Dataset of Handwritten Text Characters

Machine learning is an application of AI where the system tries to learn from its previous experience (training) and thus improve its output accuracy without being explicitly programmed. During the training of the ML algorithm, a huge data is required representing every single Input/Output class, such that ML algorithm must be aware of all possible styles of writing a single alphabet.

There are many datasets available for handwritten characters that can be used for the training phase of machine learning algorithms. One of such data is EMNIST dataset [20]. The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 [21] and converted to a 28×28 pixel image format. The EMNIST letters dataset contains $145,600$ characters with 26 classes each representing one alphabet out of 26. Each alphabet has 5600 images including small and capital letters.

The EMNIST data mentioned above is adequate for training the ML algorithm. Here requirement of a new dataset was needed, written by myself, just to increase the accuracy rate of my handwritten characters. The new data, which will be discussed in Chapter 2 contains $57,200$ images of characters, of the same size as of EMNIST dataset. Both of the data of $145,600 + 57,200 = 202,800$ characters are used for training of ML algorithm.

1.6. Literature Review

Artificial intelligence can be defined as the art of creating machines, that perform functions, which actually require intelligence when it is performed by humans [22]. AI has received significance importance in recent years with increased accuracy. Problems which used to be solved without it, are being solved or tried to be solved using different techniques of AI to achieve higher accuracy. Applications of AI include robotics, medical, industry, astronomy, agriculture, games and much more [23]. Handwritten character recognition is also one of the major application of AI.

In off-line handwritten character recognition systems, the neural networks have been successfully used to yield comparably high recognition accuracy levels. Several useful applications including mail sorting, bank processing, document reading and postal address recognition require off-line handwriting recognition systems. As a result, the off-line handwriting recognition continues to be an active area for research towards exploring the newer techniques, that would improve recognition accuracy [24, 25].

In the past decades, a large number of researcher have been dedicated to character recognition. The first driving force behind handwritten text classification was for digit classification for postal mail. Jacob Rabinows early postal readers incorporated scanning equipment and hardwired logic to recognize mono-spaced fonts [26]. Allum et. al [26] invented a system which used to sort out posts on the basis of their shipment number, which defines sender and receiver addresses.

The first prominent piece of OCR software was invented by Ray Kurzweil in 1974 which allowed for the recognition of any font [27]. This software used a more advanced form of the matrix method (pattern matching) for recognition. Essentially, this would compare bitmaps of the template character with already available bitmaps of the character and would determine to which character it is matched closely. The downside was this software was sensitive to variations in sizing and the distinctions between each individual and their way of writing.

1.6.1. Text Region detection

Rodolfo P. et al. [28] discussed a method of text line segmentation using histogram projection. The method discussed in [28], includes a couple of stages starting with binarization of the original image which is termed as feature extraction. Then a Y-histogram projection is obtained to detect possible text lines. False text lines are excluded using a threshold based on line spacings. X-histogram projection is used now on each detected text line to obtain the final result which is alphabets in those text lines.

Devadeep S. et al. [29] also used the histogram to detect text lines and to differentiate text regions from non-text regions. Text line extraction from complex layout documents is carried out using the concept of dilation and histogram [30]. Preethi [31] discussed an efficient line extraction technique from handwritten document images using the histogram and connected component analysis.

Barlas at al. [32] worked on recognizing handwritten or typed text in a heterogeneous document and developed analysis system for text recognition. This module concerned the task of document segmentation into 8 classes of homogeneous areas: text, photographic image, hand-drawn line area, graph area, table area, edge line area, separator, and material damage area. In their work they presented connected component based strategy for identification and segmentation of the text and heterogeneous documents are dealt with it, which make it different and credits to learning based approach.

Yao et al. [33] worked on multi-oriented text detection and proposed a new unified framework for it. Text detection and recognition is done all together and the same features are utilized. The system worked well on multiple orientations of the text. A new search based dictionary approach is invented to eliminate errors caused by resembling symbol [33]. Firstly, the candidates are generated via clustering and Stationary Wavelet Transform (SWT), then recognition is performed using the similar classification schema. The dataset of ICDAR2003, 2005 and 2011 are used [34].

Vaidya et al. [35] discussed a handwritten character recognition method using a convolutional neural network using the EMNIST dataset [20]. The accuracy of 94 % was obtained after training the neural network. An android application was also developed which detected alphabets and the pass the detected alphabets for recognition. A final digital text was generated in the final output.

1.6.2. Text Character Recognition

Immense research is going on in the field of handwritten character recognition. Many people have developed systems for handwritten character recognition. Some of the research work in this field is listed below:

A character recognition system has been designed using fuzzy logic [36]. System developed can be created using Very Large Scale Integration (VLSI) structure. But the system is immune to some variation and distortions in the shift. Hamming neural network is used in the system.

An innovative method for recognition of handwritten Tamil characters using neural networks has been developed [37]. Kohonen Self Organizing Map (SOM) has been used for the recognition of some Indian languages. The accuracy of this system depends a lot on accurate segmentation of handwritten characters.

Murthy et al. [38] presented a unique method for authenticating a person based on their handwriting. Multilayer feed-forward neural network has been used to authenticate a person based on the height and width of handwritten characters written by the person.

A novel method for handwritten character recognition has been designed which does not use feature extraction [39]. A feed forward neural network with back-propagation has been used in this work.

2. PROPOSED STUDIES

Text extraction is an important phase in document recognition systems. In order to detect all alphabets accurately, it is necessary to detect all possible text regions in the document. In Section 2.1, an efficient algorithm is proposed to detect all handwritten text characters from a page, written specifically, to prepare a dataset of handwritten alphabets at large scale, to be used in training phase of Machine Learning algorithms. The text line extraction algorithm uses a series of different steps to obtain the text region. Following, a sequence of histogram projection and recovery is proposed to obtain the line segmented region of the text. Text line positions are detected using a horizontal histogram projection. Vertical histogram projection is used in each individual text line to find out the positions of alphabets in the respective text line. In post-processing, noise which is mostly small black spots (similar to salt and pepper noise), is removed using a moving median filter. Histogram projections are used once again, to detect all alphabets again after removal of noise.

Handwritten text is not always the same in shape, size, formation, direction, boldness etc. All these features depend upon the writer and the environment. Section 2.1 discusses text lines detection which is straight and horizontal. But the text lines are not always straight and horizontal for all the writers. There can also be some intervention of one alphabet in two or more than two text lines. This also makes it difficult to detect the text lines accurately. Section 2.2 detects the text lines while addressing all these problems very efficiently. The algorithm starts with the estimation of skew angles by dividing original image into parts, using a mean rectangular filter at different angles and creating a filtered image at optimum skew angles of each divided part of the original image. This final image is used for text lines detection. Wrong text lines detections are removed while detecting them and at the later stage as well. The detected text lines are then used to find the alphabets in them, based on connected component logic. Those alphabets are then passed through a text recognition system which identifies each alphabet. Section 2.3 discusses a 49 layer Deep Neural Network trained for recognition of handwritten alphabets.

2.1. Preparation of Dataset

Machine learning is an application of artificial intelligence that makes a system able to automatically learn from its previous experience (training) and thus improve its output accuracy without being explicitly programmed. During the training of Machine Learning algorithm, a huge data is required representing every single Input/Output class, such that, the Machine Learning algorithm should be aware of all possible outcomes. If working on alphabets recognition in any language, writing a single alphabet 4000+ times is not an easy task.

Text line extraction is an important problem, that does not have a universal accepted solution in the context of automatic handwritten document recognition systems [40] as every writer have a different way of writings and may differ in various aspects. There have been a lot of researches in this area, and a number of algorithms have been proposed for the extraction of text lines in machine-printed document images [41, 42]. Characteristics of handwritten text can vary in font, size, shape, style, orientation, alignment, texture, colour, contrast and background. These variations turn the process of word detection to be complex and difficult [43, 44].

Rodolf P. [28] discusses text line extraction using a series of histogram projections at different orientations. A set of 1363 text lines were detected on applying the algorithm on 150 images of IAM database [45] in which 15 false text lines are detected, giving false alarm rate equal to $15/1363 = 1.1$ %.

2.1.1. Proposed Method

Dataset of handwritten characters is prepared using a technique mentioned in [28] with some alterations and focuses to ease out the whole process of preparing dataset at large scale, which later can be used to train any of Machine Learning algorithm. To prepare the dataset, the alphabets are written in a specific format, following predetermined criteria. The criteria include writing of alphabets in straight horizontal

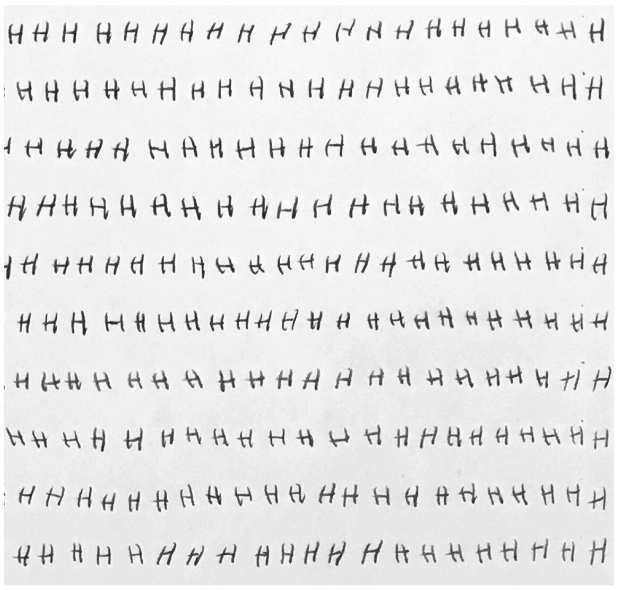


Figure 2.1. A cropped part of the scanned page of handwritten alphabet “*H*”.

lines, with significant spacings between the alphabets. While preparing these documents, an effort has been made to keep text lines straight and horizontal as much as possible with significant spacings almost equal to the height of alphabets between the text lines. A small part of one of such documents has been shown in Fig. 2.1 containing the capital letter “*H*”.

There are total 52 such pages i.e., 2 images for every single alphabet (for small and capital letters separately). Each image is named by the alphabet written in it followed by a number 1 or 2 representing whether the image contains capital or small alphabets respectively. As an example, the name “*A*1” represents an image of capital *A*s and “*h*2” contains small *h*s. Based on the name of the image, the output images of individual alphabets will be placed in a folder named after the name of the original image. All 52 documents contain around 30+ text lines and each line have around 30+ alphabets on an *A*4 size white plain paper.

The proposed algorithm starts with the binarization of the original grey-scale image, followed by 2 step histogram projections for detection of text lines and then individual characters respectively. The final step includes the removal of noise present in the

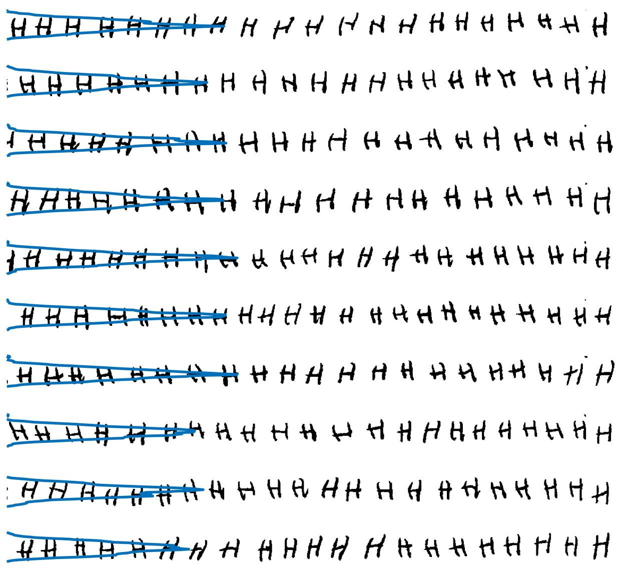


Figure 2.2. Binary image with its horizontal histogram.

scanned original image.

2.1.2. Pre-processing

Preprocessing of the original image includes removal of non-homogenous contrast in the background and binarization of the scanned grey-scale image. Both of these results can be achieved by binarization of the original image.

Binarization of an image can be represented as:

$$Im_{out} = \begin{cases} 1 & \text{for } Im_{in} \geq th. \\ 0 & \text{for } Im_{in} < th. \end{cases} \tag{2.1}$$

where Im_{in} is the input image, Im_{out} is output image and th is the threshold value.

Eq. 2.9 represents the simple definition of binarization of any greyscale image. The value of th can be set to 0.5 in the default case.

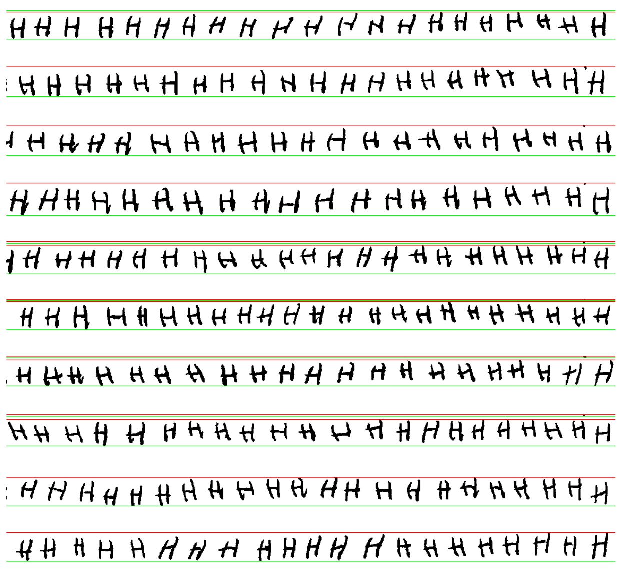


Figure 2.3. Binary image with boundaries of its text lines (red is upper boundary and green is the lower boundary of each text line).

Bradely and Roth [46, 47] presented the process of adaptive thresholding, as a form of thresholding that takes into account, the spatial variations in illumination and spatial variations that occur during scanning of documents. This technique converts the grey-scale image into a binary one, using real-time adaptive thresholding of input image. The value of the threshold for adaptive binarization in Eq. 2.9, can be calculated as the mean of neighbour values around a specific pixel in an image:

$$th(x,y) = \frac{1}{s_w \times s_h} \sum_{i=x-s_w}^{x+s_w} \sum_{j=y-s_h}^{y+s_h} Im_{in}(i,j) \tag{2.2}$$

$$s_w = 2 \times \frac{w}{16} + 1; \qquad s_h = 2 \times \frac{h}{16} + 1$$

where $th(x,y)$ is the threshold value at (x,y), $Im_{in}(i,j)$ is the intensity level of image Im_{in} at (i,j), s_w & s_h are the neighbourhood sizes (width and height respectively) for calculating average and w & h are, width and height of original image respectively.

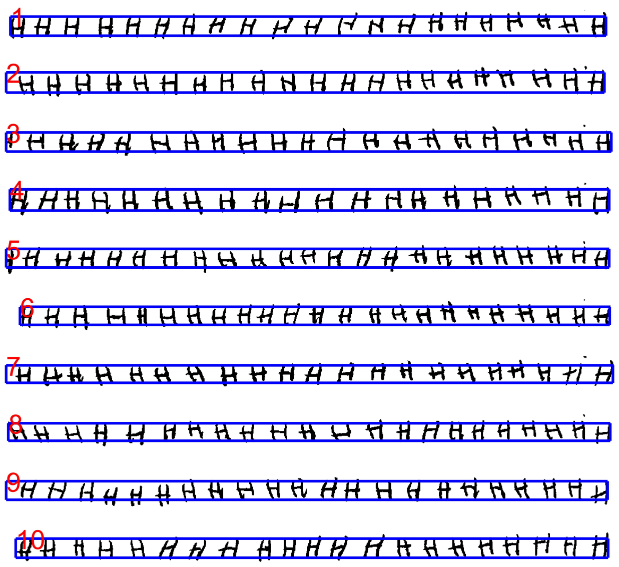


Figure 2.4. Binary image with detected text lines in blue colour and line number written on the left side of each line.

2.1.3. Text Lines Detection

For alphabets to be detected accurately, the first and most important step is to find out the boundaries of text lines accurately. As the purpose of this work is to prepare dataset of handwritten alphabets, following pre-defined criteria, the chances of any inclination of text lines are very less, as the documents are prepared very carefully.

Upper and lower boundaries of text lines can be determined using the horizontal histogram. The horizontal histogram of each row in the entire image can be represented as; $n_{horizontal} = n(1), n(2), \ldots, n(H)$, where $n(1)$ is the histogram of the first row, $n(H)$ is the histogram of the last row, and its y_{th} value (histogram of y_{th} row) can be represented as:

$$n_{horizontal}(y) = W - \sum_{x=1}^{W} I_{in}(x, y) \tag{2.3}$$

where, W is the width of the original image, H is the height of the image in pixels and

$I_{in}(x, y)$ is the intensity level of the binary image at (x, y).

Horizontal histogram of the binary image is shown on the left side in Fig. 2.2 in blue colour. Horizontal histogram with 0 value represents blank spacings between the original text lines. Using this histogram, the text lines where the histogram is 0, are marked with different colour as shown in Fig. 2.3. Some extra text lines (between 4^{th} and 8^{th} actual text lines) can be seen on careful examination of Fig. 2.3. These wrong detections are due to the fact that the histogram is 0 at some points, due to the presence of some small dots corresponding to it. These small dots will be removed in post-processing of the algorithm.

The text lines which are wrong detections are small in vertical height compared to all other text lines. A sorting algorithm is applied, based on vertical widths of text lines, in which text lines having widths more than mean width of all text lines, are removed. The text lines after the removal of falsely detected lines are shown in Fig. 2.4.

2.1.4. Alphabets Detection

The next step after the successful detection of text lines is accurate detection of alphabets. The technique used here is the same as it is used for text lines detection. Alphabets are detected using vertical histogram projection in all text lines individually.

The vertical histogram of each individual column, in a text line, is represented as; $n_{vertical} = n(1),\ n(2),\ . . . ,\ n(W)$, where $n(1)$ is the histogram of the first column, $n(W)$ is the histogram of the last column of the text line, and its x_{th} value (histogram of x_{th} column) is represented by:

$$n_{vertical}\left(x\right) \;=\; H_t - \sum_{y=1}^{H_t} I_t\left(x, y\right) \qquad (2.4)$$

where, H_t is the height of specific individual text line and $I_t(x, y)$ is the intensity level at (x, y) in that specific text line.

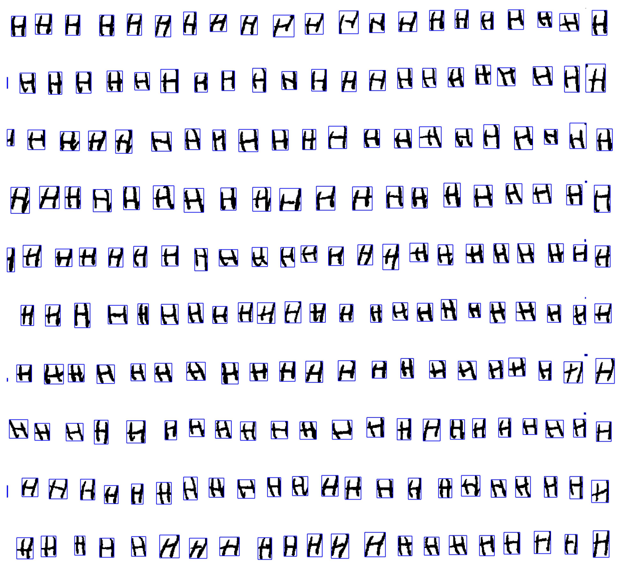


Figure 2.5. Binary image with objects detected and marked with blue boundaries.

The value of vertical histogram is 0, at the points where there is blank space between the alphabets. These values of vertical histogram projection are used to find out the horizontal positions of all alphabets. Detected alphabets are shown in blue rectangular boxes in Fig. 2.5.

2.1.5. Post-Processing

The purpose of this post-processing is to remove all non-alphabetical objects from Fig. 2.5. As the alphabets are written very carefully, the only possibility of non-alphabetical objects is some small dots, present on the paper while writing. Some of those small dots can also be clearly seen in Fig. 2.5.

An intermediate image is generated, such that, each detected object is transformed into rectangular black objects of the same size, as it is in blue boundaries, shown in Fig. 2.5. This results in the small dots close to alphabets, to be classified as part of the respective alphabet. Because chances of small dots, written close to the alphabets mistakenly, are very less.

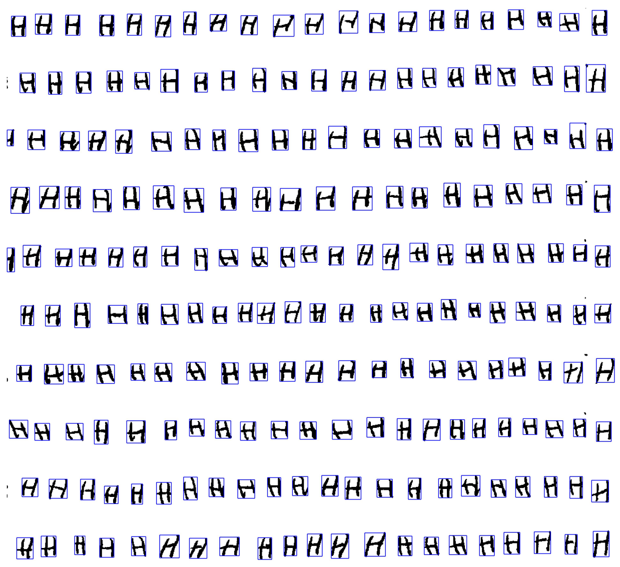


Figure 2.6. Binary image with objects detected after removing small dots and marked with blue boundaries.

A moving median filter is applied on the intermediate image resulting removal of small dots, which were not part of alphabets. Size of the median filter depends upon the mean size of all detected objects.

Fig. 2.6 shows the final objects detected after removal of small dots from Fig. 2.5. Comparing both Figures, the difference in detected alphabets can be noticed, i.e. small dots, which were classified as alphabetical objects are now removed.

2.1.6. Experimental Results

There is no other research carried out on the preparation of handwritten alphabets. So in that term, this is the first paper which discusses, the method to prepare a dataset of handwritten characters and to ease out the whole process. The discussed algorithm was applied to the scanned documents mentioned in section 2.1.1. The total objects detected from all 52 scanned documents are 71180, out of which 70411 objects were detected correctly. Detection rate for these experiments resulted to be 98.9 % and false detection rate to be 1.1 %.

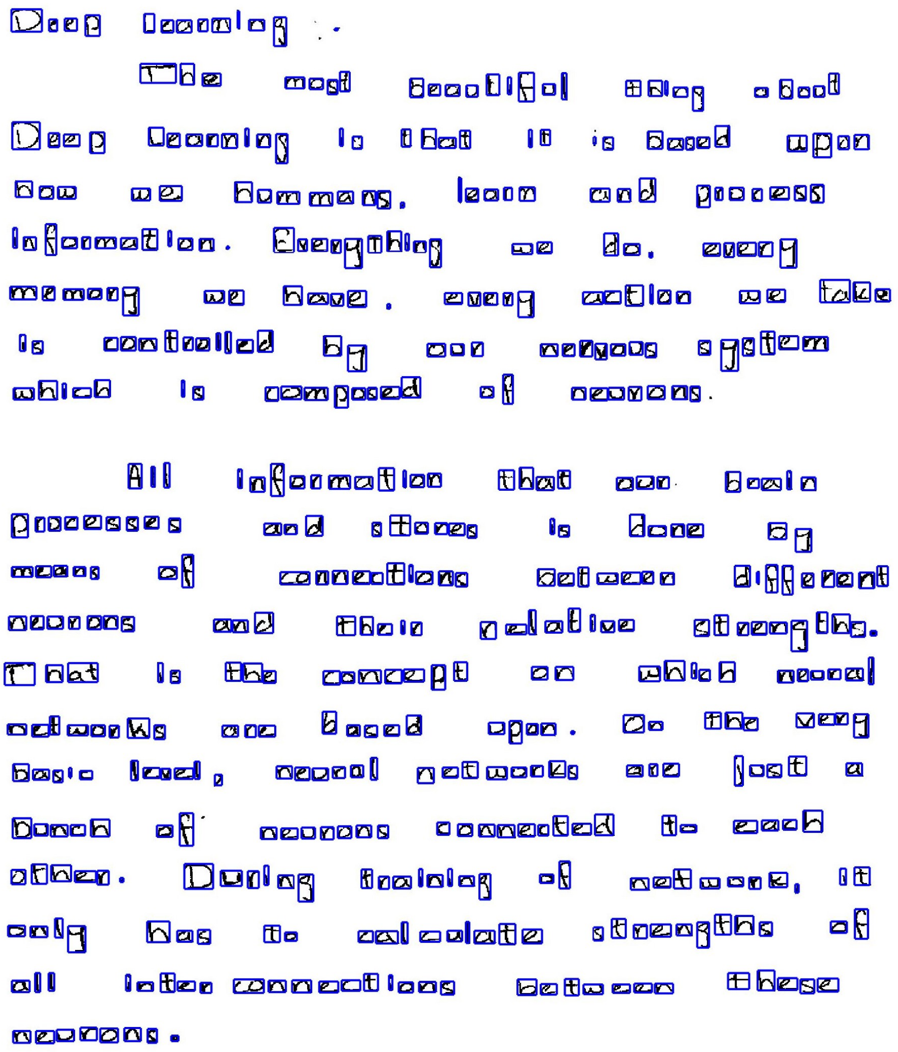
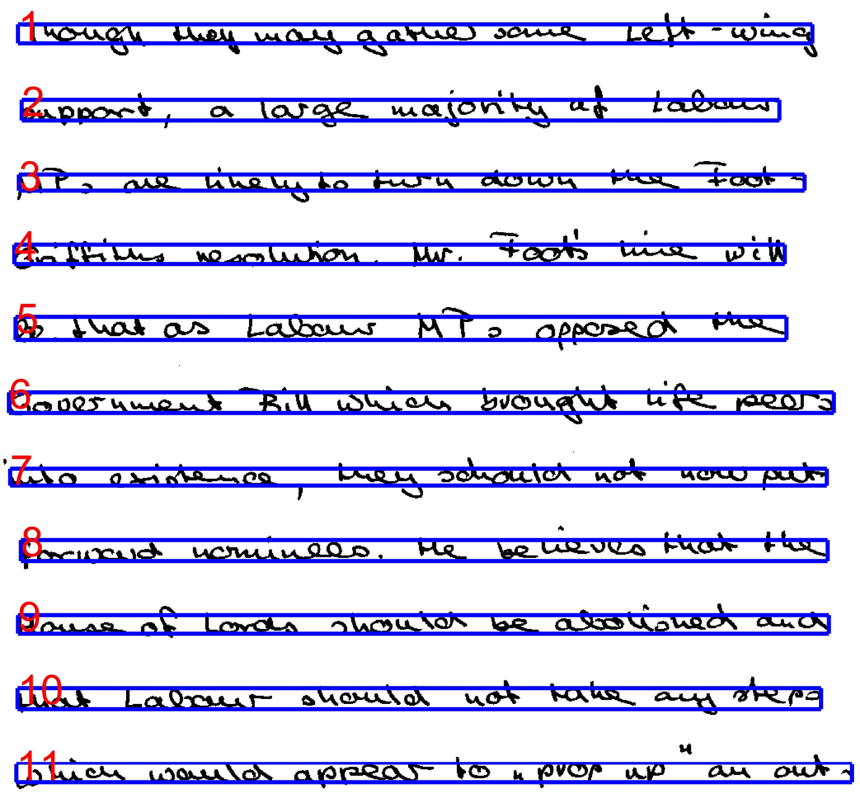


Figure 2.7. Objects detected in a normal handwritten document.

Fig. 2.7 represents a sample text specifically prepared for the proposed study. The total number of alphabets in this text document are 543 and all these alphabets are detected accurately except some non-alphabetical objects detection. The total number of those non-alphabetical objects are 11, which yields 98 % of detection rate and 2 % false detection rate.

The same algorithm mentioned in Section 2.1.3 is also applied on 100 images of IAM dataset [45] for detection of text lines, just to compare results with [28]. Fig 2.8 shows the output of one of those images. The total number of text lines detected in these 100 images are 945 out of which 940 are detected accurately giving false detection rate equal to $(945 - 940)/945 = 0.53$ %. Whereas, the false detection rate for [28] is



1 though they may gather some Left-wing
2 support, a large majority of Labour
3 MPs are likely to turn down the Foot-
4 Griffiths resolution. Mr. Foot's line will
5 be that as Labour MPs opposed the
6 Government Bill which brought life peers
7 into existence, they should not now put
8 forward nominees. He believes that the
9 House of Lords should be abolished and
10 that Labour should not take any steps
11 which would appear to "prop up" an out-

Figure 2.8. An image from IAM database with text lines detected.

1.1 %, proving better results of our proposed algorithm also shown in Table 2.1.

2.1.7. Conclusion

In conclusion, a novel algorithm is proposed for detection of text lines and handwritten characters, written in the specified format discussed in section 2.1.1. According to the research carried out, there is no other paper which presents a method to prepare a dataset of handwritten characters or to ease out this process. This method eases and speeds up the process of preparing large dataset required for training of Machine Learning algorithms. Difference between the proposed method and one in [28] is an addition of extra step i.e., thresholding text lines on the basis of their vertical widths, mentioned in section 2.1.3.

The proposed algorithm is the first phase for the solution of automatic handwritten document recognition systems, where the criteria of writing characters in straight hor-

Table 2.1. Detection rates of different experiments. Detection rates of different experiments. Detection rates of different experiments. Detection rates of different experiments.

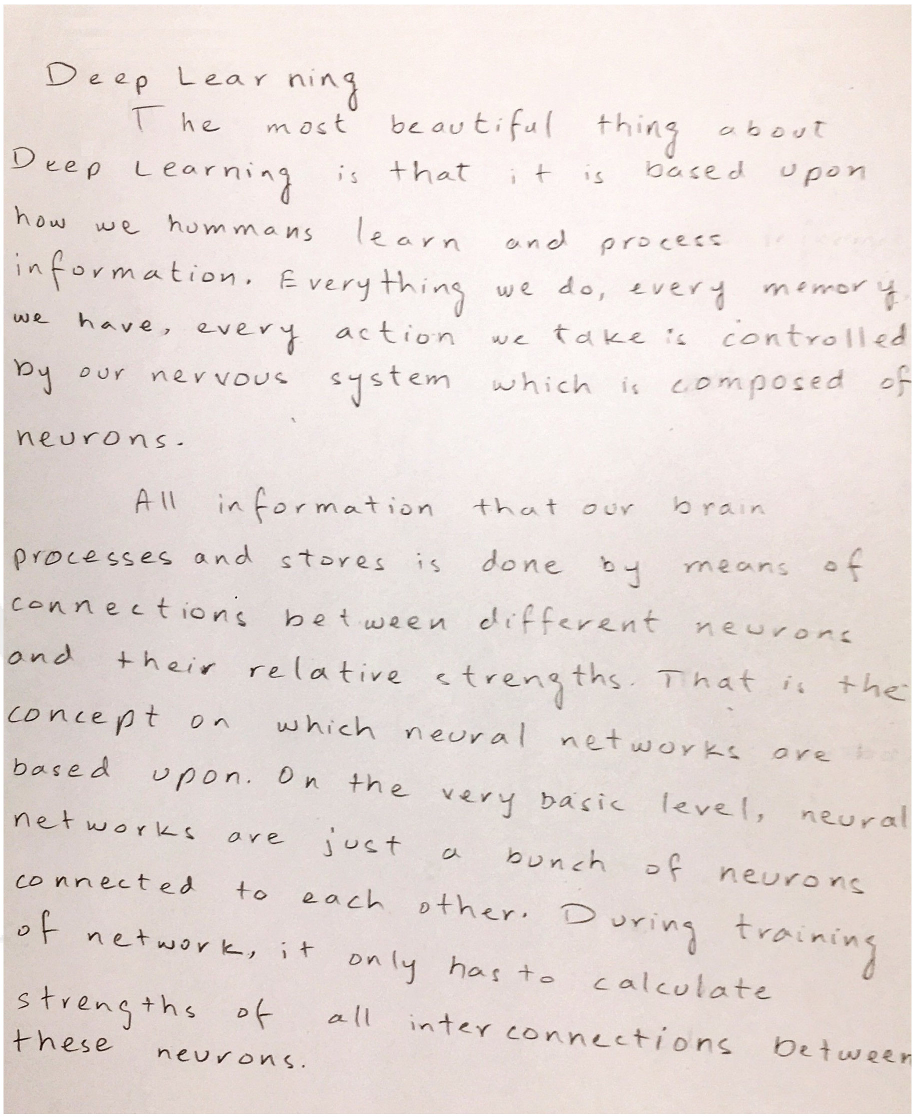
	Proposed	Rodolf P. [28]
Text line detection on IAM database	99.47 %	98.9 %
Alphabets detection for dataset preparation	98.9 %	–
Alphabets detection for normal document	98 %	–

izontal lines, without overlapping of text lines with each other, on a white plain paper, has to be followed.

The next phases of this study, are the compensation of inclined text lines at any angle and removal of figures & plots drawn in between handwritten text documents. So that class notes, written by a student during their lectures, can also be translated into digital copies. The technique discussed here can be used with some advancements for these problems as well.

2.2. Text Lines Detection

For a text to be recognized, the first and the most important step is the accurate detection of text lines in a handwritten document. While writing, most of the writers do not focus on the straightness of there text lines. A writer may start writing a document carefully, intending for the text lines to be straight and horizontal, but it might end with some inclination growing in each text line. Fig. 2.9 shows a handwritten document, where the writer starts carefully, but even then the text lines get some inclination in downwards direction. The first line in Fig. 2.9 looks to be at around $0°$ but the last lines are at about $4° - 6°$ degrees compared to the first line. Intertext line spacings are also not constant for all the text lines. There are some alphabets which can be



Deep Lear ning
The most beautiful thing about
Deep Learning is that it is based upon
how we hommans learn and process
information. Everything we do, every memory
we have, every action we take is controlled
by our nervous system which is composed of
neurons.

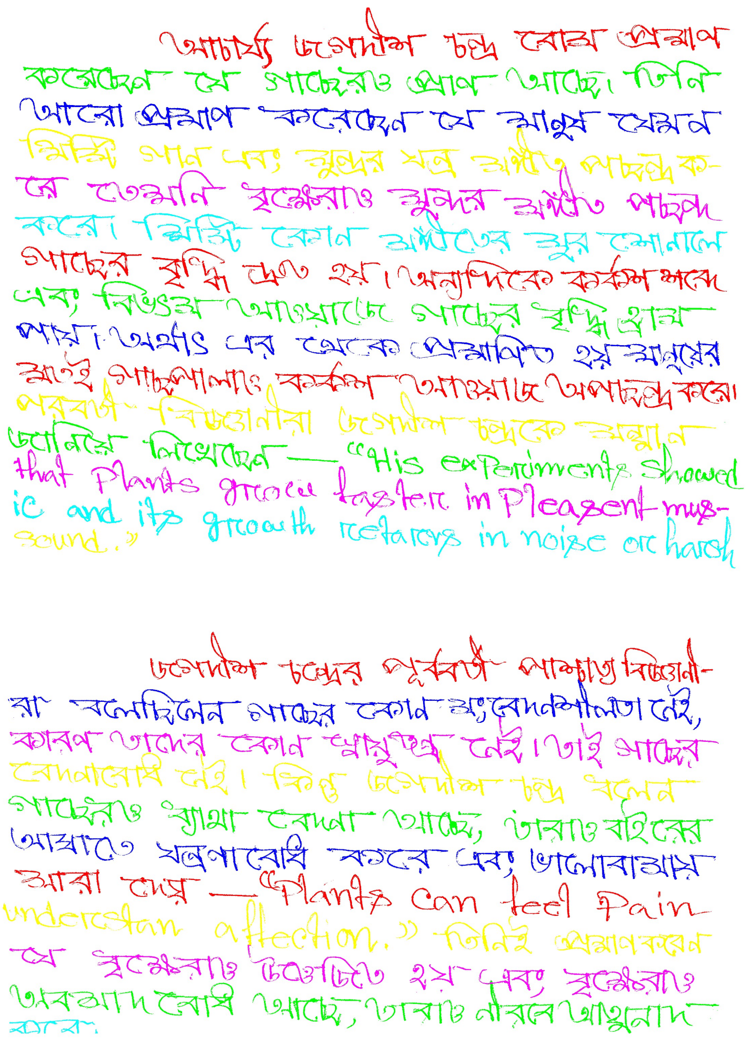
All information that our brain
processes and stores is done by means of
connections between different neurons
and their relative strengths. That is the
concept on which neural networks are
based upon. On the very basic level, neural
networks are just a bunch of neurons
connected to each other. During training
of network, it only has to calculate
strengths of all interconnections between
these neurons.

Figure 2.9. A sample of a scanned document.

classified correctly by the human but can be classified into two text lines by the system. First and second text lines are the example of such possible errors.

Section 2.1 discusses the text lines detection without any chance of inclinations in it using a series of histogram projections. Horizontal histogram technique can be tested on Fig. 2.9 for the text lines detection but it cannot give the desired text lines. It will merge the text lines which are at some inclinations, to single text line as histogram will not be 0 at points, where the text lines are at some angle or alphabets are merged into other text lines. It may work for couple of first text lines but not the later ones.

Fig. 2.10 shows another example of handwritten text in Bangla language [16, 17].


আচার্য্য জগদীশ চন্দ্র বোস প্রমাণ
করেছেন যে গাছেরও প্রাণ আছে। তিনি
আরো প্রমাণ করেছেন যে মানুষ যেমন
মিষ্টি গান এবং সুন্দর যন্ত্র সংগীত পছন্দ ক-
রে তেমনি বৃক্ষেরাও সুন্দর সংগীত পছন্দ
করে। মিষ্টি কোন সংগীতের সুর শোনালে
গাছের বৃদ্ধি দ্রুত হয়। অন্যদিকে কর্কশ শব্দে
এবং বিশ্রী আওয়াজে গাছের বৃদ্ধি হ্রাস
পায়। অর্থাৎ এর থেকে প্রমাণিত হয় মানুষের
মতই গাছপালাও কর্কশ আওয়াজ অপছন্দ করে।
পরবর্তী বিজ্ঞানীরা জগদীশ চন্দ্রকে সম্মান
জানিয়ে লিখেছেন — "His experiments showed
that Plants grow faster in Pleasent mus-
ic and its growth retards in noise or harsh
sound."

জগদীশ চন্দ্রের পূর্ববর্তী পাশ্চাত্য বিজ্ঞানী-
রা বলেছিলেন গাছের কোন সংবেদনশীলতা নেই,
কারণ তাদের কোন স্নায়ুতন্ত্র নেই। তাই গাছের
বেদনাবোধ নেই। কিন্তু জগদীশ চন্দ্র বলেন
গাছেরও ব্যাথা বেদনা আছে, তারাও বাইরের
আঘাতে যন্ত্রণাবোধ করে এবং ভালোবাসায়
সাড়া দেয় — "Plants can feel Pain
understand affection." তিনিই প্রমাণ করেন
যে বৃক্ষেরাও উত্তেজিত হয় এবং বৃক্ষেরাও
অবসাদবোধ আছে, তারাও নীরবে আর্তনাদ
করে।

Figure 2.10. A sample of a scanned document taken from CMATER database.

The second problem other than the inclination of text lines is the intersection of some alphabets with text line above or below it. This problem makes it difficult to differentiate the two text lines at first, and if the text lines are detected correctly, it makes the system difficult to label the alphabet, to which text line it actually belongs to. Such alphabets can be easily seen in Fig. 2.10.

This section focuses on the solution of text lines detection that is at some inclination and the text lines that contains overlapping alphabets.

2.2.1. Pre-processing of Scanned Document

Preprocessing of Scanned document depends upon the type of paper used and non-homogeneous contrast in the background which is mostly caused by the scanning process of documents. There are several types of papers that are used for preparing handwritten notes, i.e., a white plain paper or a paper with grids on it. A clear difference can be seen between the background of Fig. 2.9 and Fig. 2.10. Fig. 2.10 is apparently preprocessed document whereas, the background of Fig. 2.10 can be noticed with changing between white colour and greyish white colour at some point.

There are several types of pre-processing algorithms which can be used depending upon the types of documents. Some of those which are used in this research are mentioned below:

Bottom - Hat filtering

A bottom - hat filter enhances black spots in a white background. It subtracts the morphological "Close" of the image from the image. The effect is to fill holes and join nearby objects. In mathematical morphology and digital image processing, bottom - hat transform is an operation that helps to highlight the dark spots in given images. Extract small, dark regions from an image. The bottom - hat transform effectively inverts high-frequency regions. Thus it also suppresses the light grey coloured background grid. bottom - hat transforms are used for various image processing tasks, such as feature extraction, background equalization, image enhancement, and others.

Bottom - Hat filtering can be represented as [48]:

$$B_{hat}(f) = (f \bullet b) - b \tag{2.5}$$

where, f is an image on which bottom - hat filtering is to be applied, b is a structuring element and $\bullet$ is a closing operation.

Closing operation ($\bullet$) of image f by structuring element b is defined as the erosion of f by b followed by a dilation of the result with b [48]:

$$(f \bullet b) \;=\; (f \oplus b) \ominus b \tag{2.6}$$

where, $\oplus$ and $\ominus$ denote dilation and erosion, respectively.

The erosion of f by a flat structuring element b at any location (x, y) is defined as the minimum value of the image in the region coincident with b when the origin of b is at (x, y). Therefore, the erosion at (x, y) of an image f by a structuring element b is given by [48]:

$$[f \ominus b]\,(x, y) \;=\; min_{(s,t)\epsilon b}\,[f(x + s, y + t)] \tag{2.7}$$

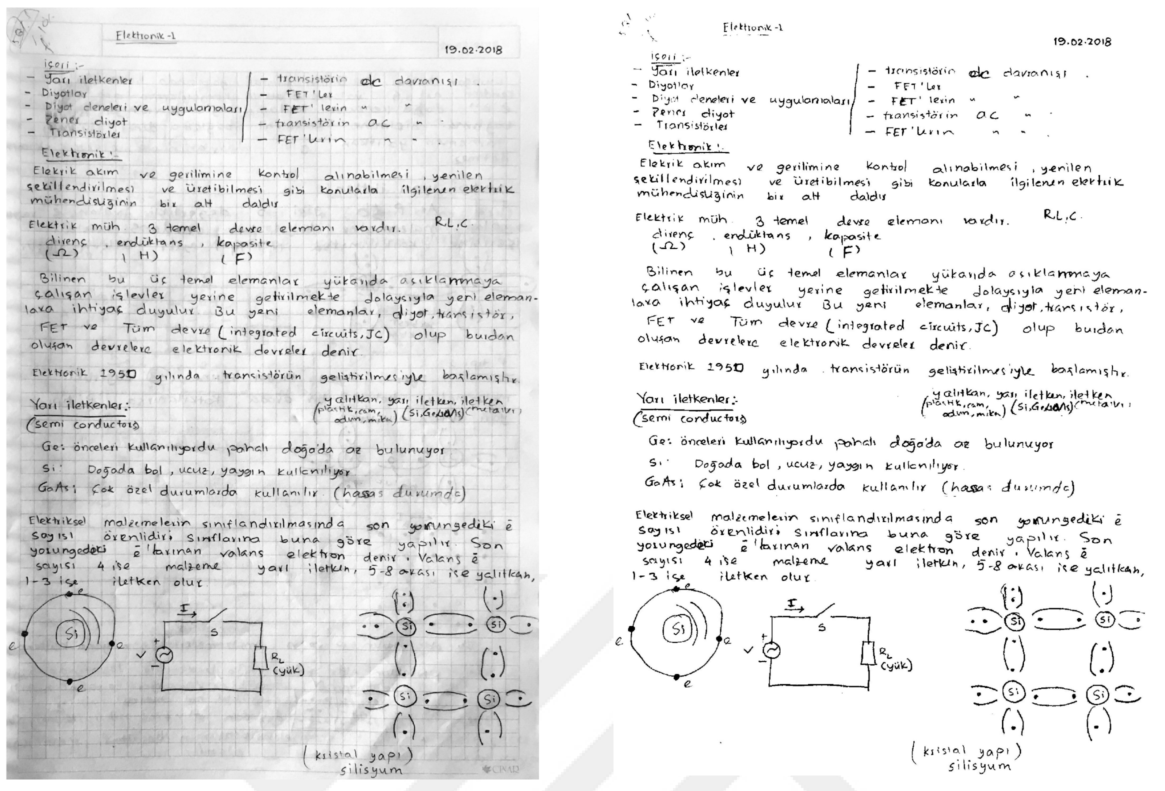
The explanation is similar to one for erosion except for using maximum instead of a minimum. The dilation of f by a flat structuring element b at any location (x, y) is defined as the maximum value of the image in the window outlined by b, when the origin of b is at (x, y) [48]. That is:

$$[f \oplus b]\,(x, y) \;=\; max_{(s,t)\epsilon b}\,[f(x + s, y + t)] \tag{2.8}$$

Fig. 2.11b shows the result of bottom - hat filtering applied on Fig. 2.11a. The original image contains grids at its background. Most of this grid background shades out during bottom - hat filtering, Rest of the shaded grids are removed during binarization of this filtered image.

Histogram Equalization

Fig. 2.12a shows a sample image taken from CMATER database [16, 17] in the Bangla language. The database is published in colourful form such that each text line is represented with different colours. While converting an image of such type into greyscale or binary form, some of colours just totally disappear or fade out. Averaging the RGB colours also does not work in this case, as binarization fades out the light



Elektronik-1

19.02.2018

İçeri :-

- Yarı iletkenler
- Diyotlar
- Diyot deneleri ve uygulamaları
- Zener diyot
- Transistörler
- transistörün dc davranışı .
- FET'ler
- FET' lerin " "
- transistörün ac " .
- FET'lerin " " .

Elektronik :

Elekrik akım ve gerilimine kontrol alınabilmesi , yenilen şekillendirilmesi ve üretibilmesi gibi konularla ilgilenen elektrik mühendisliğinin bir alt daldır

Elektrik müh. 3 temel devre elemanı vardır. R,L,C.
direnç , endüktans , kapasite
(Ω) (H) (F)

Bilinen bu üç temel elemanlar yükarıda açıklanmaya çalışan işlevler yerine getirilmekte dolaysıyla yeni elemanlara ihtiyaç duyulur Bu yeni elemanlar, diyot, transistör, FET ve Tüm devre (integrated circuits, JC) olup bundan oluşan devrelere elektronik devreler denir.

Elektronik 1950 yılında transistörün geliştirilmesiyle başlamıştır.

Yarı iletkenler:
(semi conductors)

yalıtkan, yarı iletken, iletken
(plastik, cam, odun, mika) (Si, Ge, GaAs) (metaller)

Ge: önceleri kullanılıyordu pahalı doğa'da az bulunuyor

Si: Doğada bol , ucuz, yaygın kullanılıyor.

GaAs: Çok özel durumlarda kullanılır (hassas durumda)

Elektriksel malzemelerin sınıflandırılmasında son yörüngedeki ē sayısı önemlidir sınıflarına buna göre yapılır. Son yörüngedeki ē'lerinen valans elektron denir. Valans ē sayısı 4 ise malzeme yarı iletken, 5-8 arası ise yalıtkan, 1-3 ise iletken olur.

(a) (b)

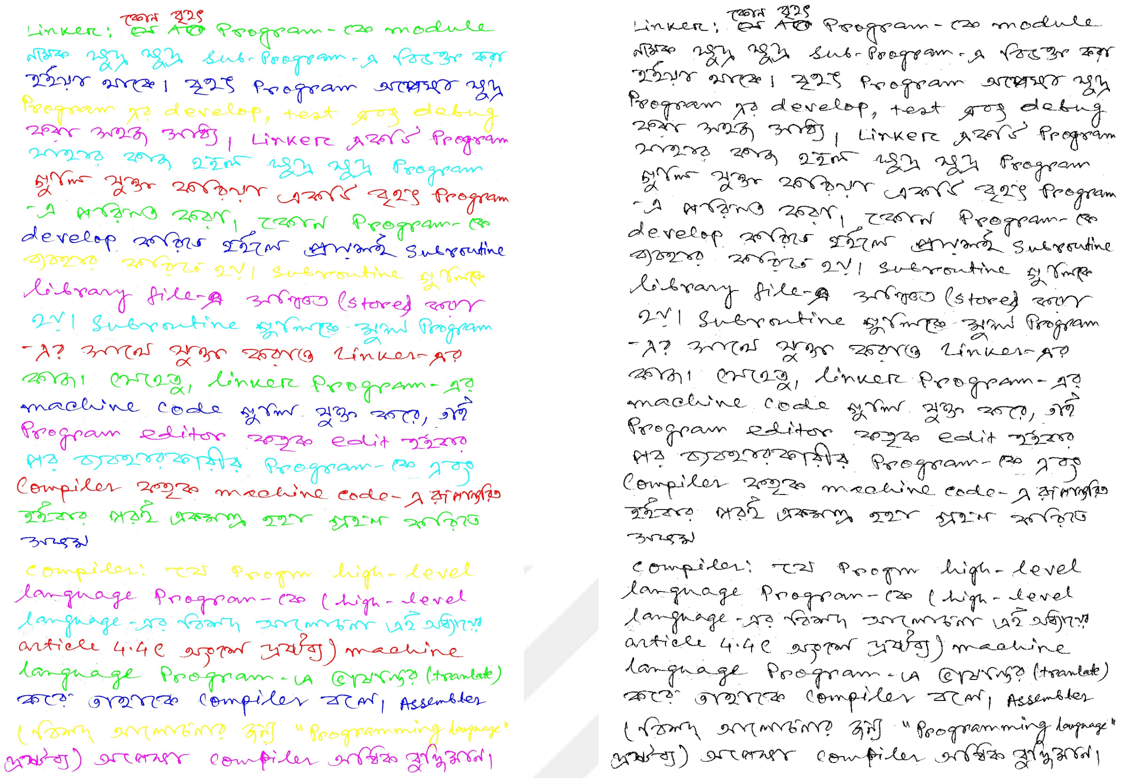
Figure 2.11. (a) Original Scanned image with the grid background and (b) bottom - hat filtering and binarization.

colours later on.

Adaptive Binarization

A document prepared on a white plain paper can be easy to process, but scanning of documents is another factor which can affect the results. Fig. 2.13a shows a white plain paper with alphabets "*A*" written on it. The document is prepared on white plain paper, but the process of scanning left a non-homogeneous contrast in the background, which can be worse than this. The background of Fig. 2.13a is no more pure white, instead it is greyish white somewhere and pure white at some points. Adaptive Binarization removes such effects while binarizing the original image.

Binarization of an image can be represented as:



(a) (b)

Figure 2.12. (a) Original Scanned image (Bangla Language) and (b) Results of Histogram Equalization and binarization.

$$Im_{out} = \begin{cases} 1 & \text{for } Im_{in} \geq th. \\ 0 & \text{for } Im_{in} < th. \end{cases} \tag{2.9}$$

where Im_{in} is the input image, Im_{out} is output image and th is the threshold value.

Eq. 2.9 represents the simple definition of binarization of any greyscale image. The value of th can be set to 0.5 in the default case.

Bradely and Roth [46, 47] presented the process of adaptive thresholding, as a form of thresholding that takes into account, the spatial variations in illumination and spatial variations that occur during scanning of documents. This technique converts the grey-scale image into a binary one, using real-time adaptive thresholding of input image. The value of the threshold for adaptive binarization in Eq. 2.9, can be calculated

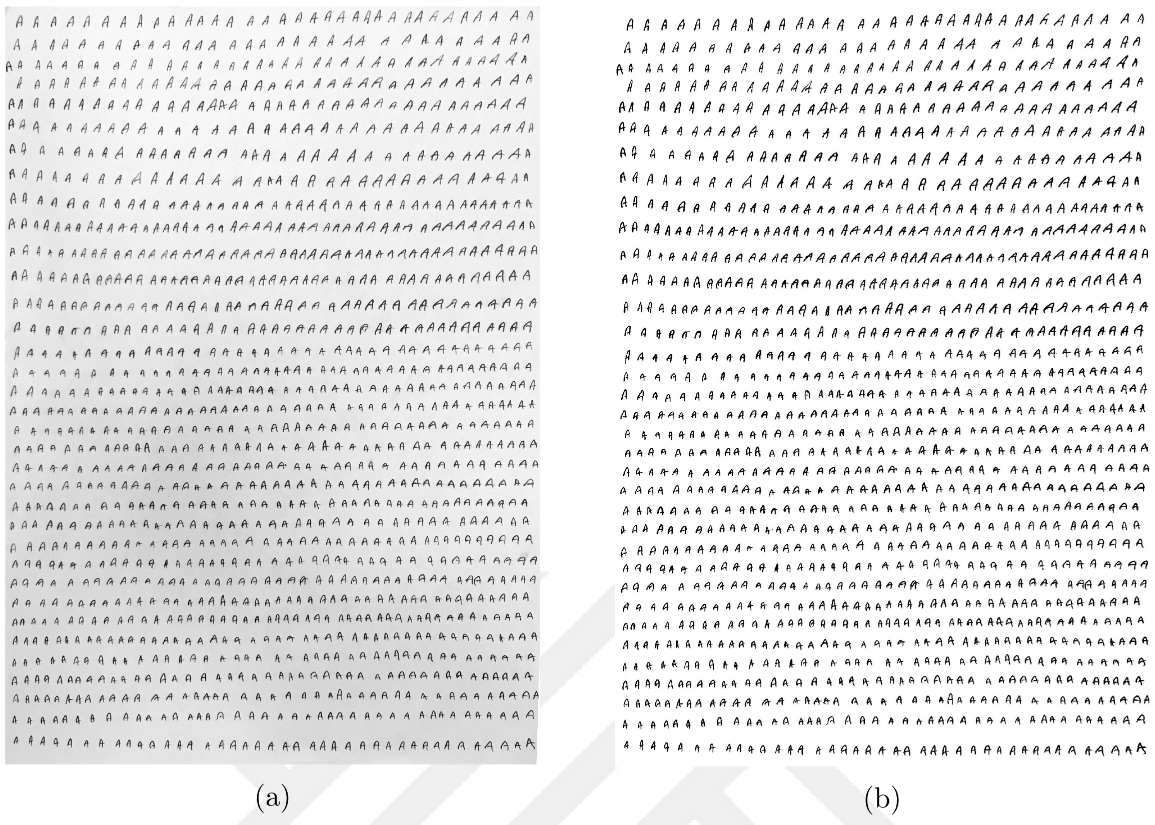


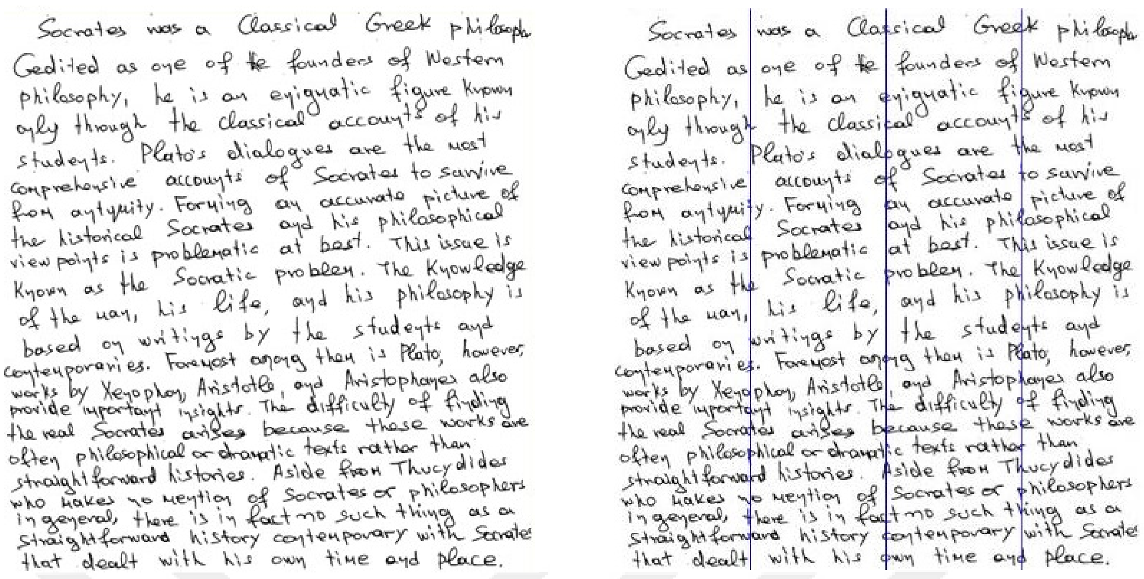
Figure 2.13. (a) Original Scanned image and (b) After adaptive binarization.

as the mean of neighbour values around a specific pixel in an image:

$$th(x,y) = \frac{1}{s_w \times s_h} \sum_{i=x-s_w}^{x+s_w} \sum_{j=y-s_h}^{y+s_h} Im_{in}(i,j) \tag{2.10}$$

$$s_w = 2 \times \frac{w}{16} + 1; \qquad s_h = 2 \times \frac{h}{16} + 1$$

where $th(x,y)$ is the threshold value at (x,y), $Im_{in}(i,j)$ is the intensity level of image Im_{in} at (i,j), s_w & s_h are the neighbourhood sizes (width and height respectively) for calculating average and w & h are, width and height of original image respectively.


Socrates was a Classical Greek philosoph
Credited as one of the founders of Western
philosophy, he is an enigmatic figure known
only through the classical accounts of his
students. Plato's dialogues are the most
comprehensive accounts of Socrates to survive
from antiquity. Forming an accurate picture of
the historical Socrates and his philosophical
viewpoints is problematic at best. This issue is
known as the Socratic problem. The knowledge
of the man, his life, and his philosophy is
based on writings by the students and
contemporaries. Foremost among them is Plato; however,
works by Xenophon, Aristotle, and Aristophanes also
provide important insights. The difficulty of finding
the real Socrates arises because these works are
often philosophical or dramatic texts rather than
straightforward histories. Aside from Thucydides
who makes no mention of Socrates or philosophers
in general, there is in fact no such thing as a
straightforward history contemporary with Socrate
that dealt with his own time and place.

(a) (b)

Figure 2.14. (a) A sample of scanned document for text line extraction and (b) Scanned document divided vertically into 4 equal parts.

2.2.2. Estimation of Skew Angles

Considering Fig. 2.14a [49] as the next input, the text lines are not horizontal and straight, and the inter text lines spacing are also very less at the bottom half of the document. At some point, the line starts at a specific angle but it changes it as it goes further. This makes some of the text lines curvy. Some of the curvy text lines can be noticed by careful examination of Fig. 2.14a. Such text lines are difficult to trace out.

A series of different processes are applied on scanned image to estimate the skew angle of the text lines accurately, which later will be used for text lines detection.

Dividing Image Vertically

There are some writers who may start writing a text line straight and horizontal, but the text line may end at some angle which was not in the start of the text line. Such text lines can also be seen in Fig. 2.14a. The skew angle does not remain the same throughout the text line. To address this problem, the original image is divided into parts vertically, such that, the angle of text line in a single vertical part remains the same. A divided image in 4 parts is also shown in Fig. 2.14b.

Generating Filtered Images for all Angles

The text lines in Fig. 2.14a are not explicitly differentiated from each other. Some of the alphabets which belong to a specific text line might be overlapped by the text line above or below it. The original image is filtered using a rectangular averaging filter at specific angles.

Result of averaging filter at (x, y) at an angle θ is represented by [50, 51]:

$$Im_{out}(x,y) \ = \ \frac{1}{m \ \times \ n} \sum_{i=x}^{m+x} \sum_{j=y}^{n+y} Im_{in}(i\cos(\theta)-j\sin(\theta), i\sin(\theta)+j\cos(\theta)) \quad (2.11)$$

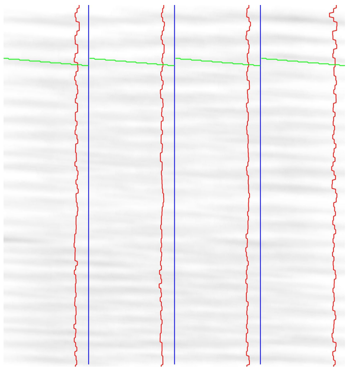
where, m & n are widths and heights of rectangular filter, Im_{in} & Im_{out} are the input and output images respectively.

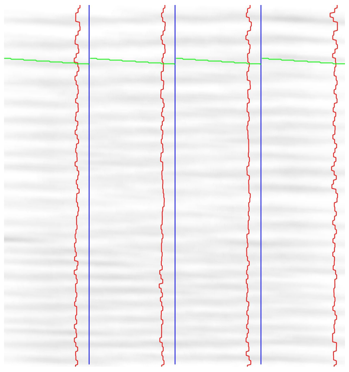
Filtered images are also shown in Fig. 2.15. All the filtered images are then divided into four equal parts, with boundaries shown in blue colour.

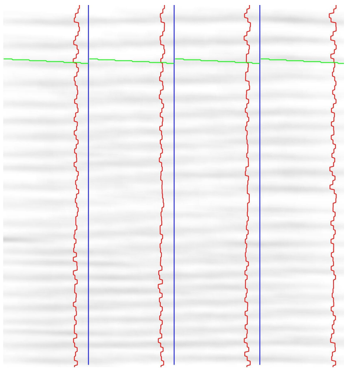
Mean of each individual vertical part is calculated at respective angles and is plotted in red at the left side of each vertical part. Mean can also be represented by using Eq. 2.11 with small modifications. Mean value of a group of rows starting from row x at angle θ, in a vertical part, can be defined as:

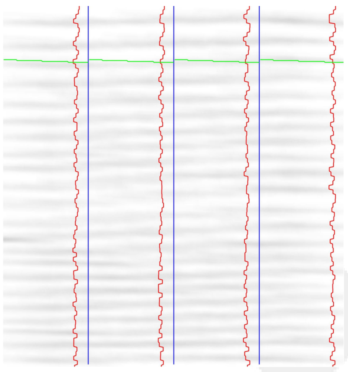
$$Me(x : x + h_p) \ = \ \frac{1}{w_p} \sum_{i=x}^{x+h_p} \sum_{j=1}^{w_p} Im_p(i\cos(\theta) - j\sin(\theta), i\sin(\theta) + j\cos(\theta)) \quad (2.12)$$

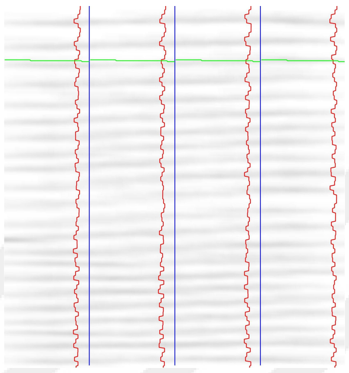
where, Im_p represents each vertical part individually, w_p is the width of Im_p and width of the rectangular window to calculate mean and h_p is the height of the rectangular window to calculate mean.

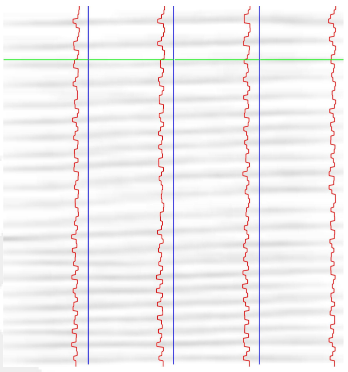

(a) At angle $-5°$

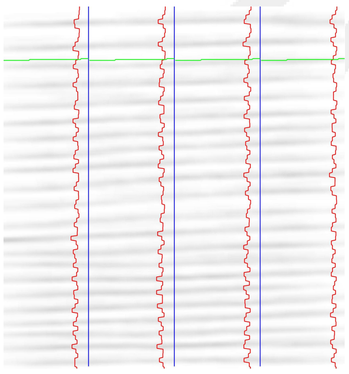

(b) At angle $-4°$

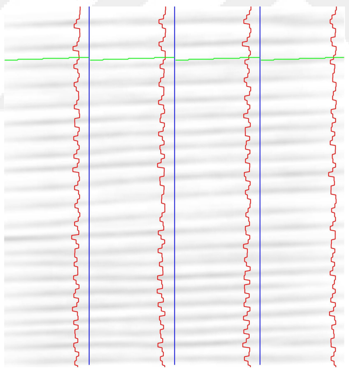

(c) At angle $-3°$

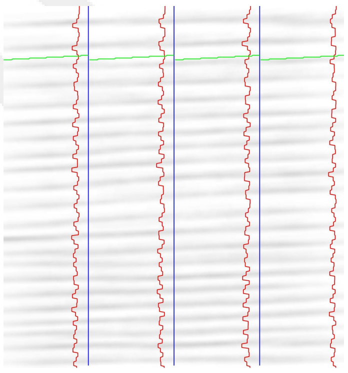

(d) At angle $-2°$

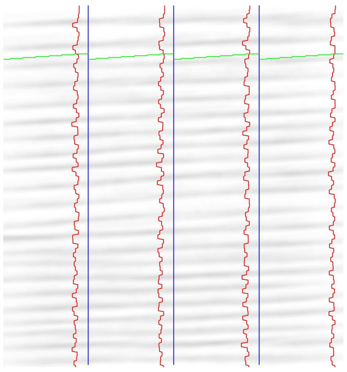

(e) At angle $-1°$


(f) At angle $0°$


(g) At angle $+1°$


(h) At angle $+2°$


(i) At angle $+3°$


(j) At angle $+4°$

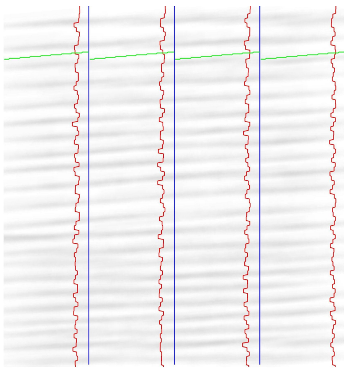

(k) At angle $+5°$

Figure 2.15. Filtered images by using the rectangular averaging filter at angles from $-5°$ to $+5°$. Vertical blue lines represent the division of original image, plots in red colour is the mean calculated horizontally of each vertical part at respective angles and green line represent a line at the respective angle for reference.

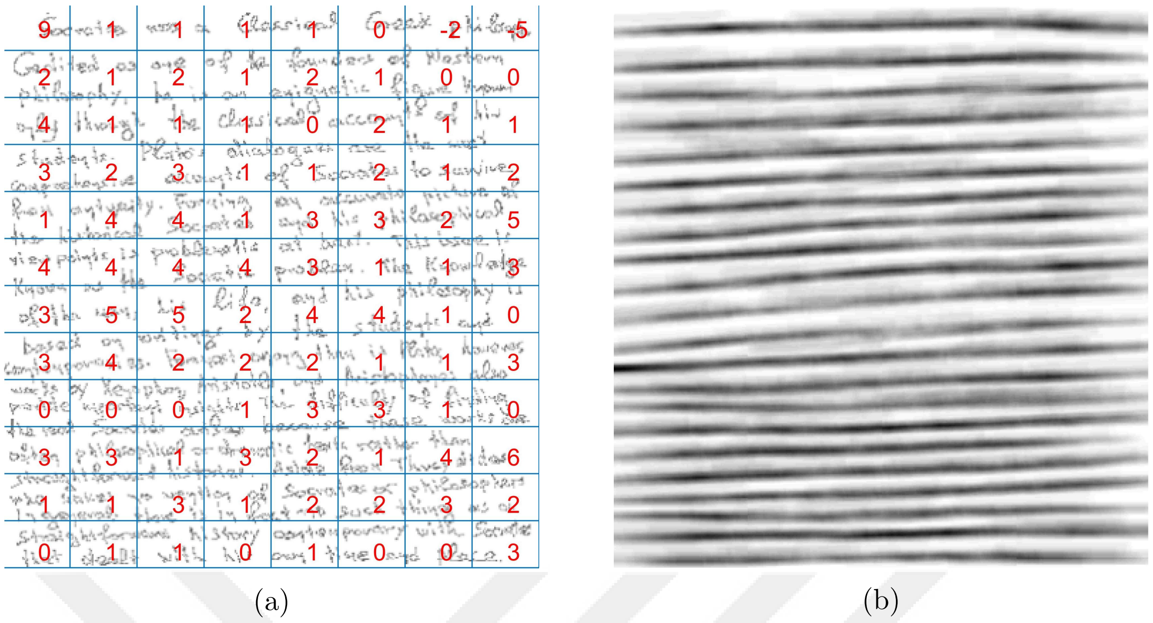


Figure 2.16. (a) Optimum angles of each part and (b) Filtered image at optimum angles.

Mean value gives the contribution of individual pixel intensity for the entire image. A mean value will be lower compared to the brighter part of an image. This concept of the mean value is used in the estimation of skew angles. Careful focus on Fig. 2.15 reveals that the mean value variation is very small at the angles other than the optimum. By comparing the mean plots of leftmost vertical parts in Fig. 2.15a and Fig. 2.15g, it is evident that variations are very small in Fig. 2.15a whereas, mean variations are comparatively large in Fig. 2.15g. This technique can be used in correct skew angles estimation.

The value of h_p in Eq. 2.12 and n, the height of rectangular filter, in Eq. 2.11 effects the final outcome. Both of these variables can be adjusted to the same value or can be close to each other, and can be changed according to the inter text lines spacings. If the text lines are much closer to each other, both of these values should be moved to lower ones, as these will cause the closer text lines to be merged into each other.

The value of m in Eq. 2.11 depends upon the inter-word spacings. Default value of m, which is used here, is the 4th part of the width of the original image.

The filtered images which are shown in Fig. 2.15, are divided into 4 vertical parts but the original algorithm uses more than 4 vertical parts. 4 vertical parts are used for a clear demonstration of filtered images (at this stage only) and their later processes.

The range of angles shown in Fig. 2.15 is from $-5^{\cap}$ to $+5^{\cap}$. The ranges here are used (at this stage only) for a clear demonstration of the algorithm, and to show all the possible filtered images on one page. The original angle range was $-7^{\cap}$ to $+7^{\cap}$.

Dividing Filtered Images Horizontally

The skew angle cannot be constant throughout one vertically divided part. The first text line may be at a different angle than the next one. For this reason, the images in Fig. 2.15 are divided horizontally as shown in Fig. 2.16a. Likewise, mean values plotted in Fig. 2.15 are also divided horizontally.

The final divided image as shown in Fig. 2.16a is divided into 12 $\times$ 8 horizontal and vertical parts respectively. Each of these parts has there own mean values representing the rows. These mean values are used for skew angles estimation for the respective divided part.

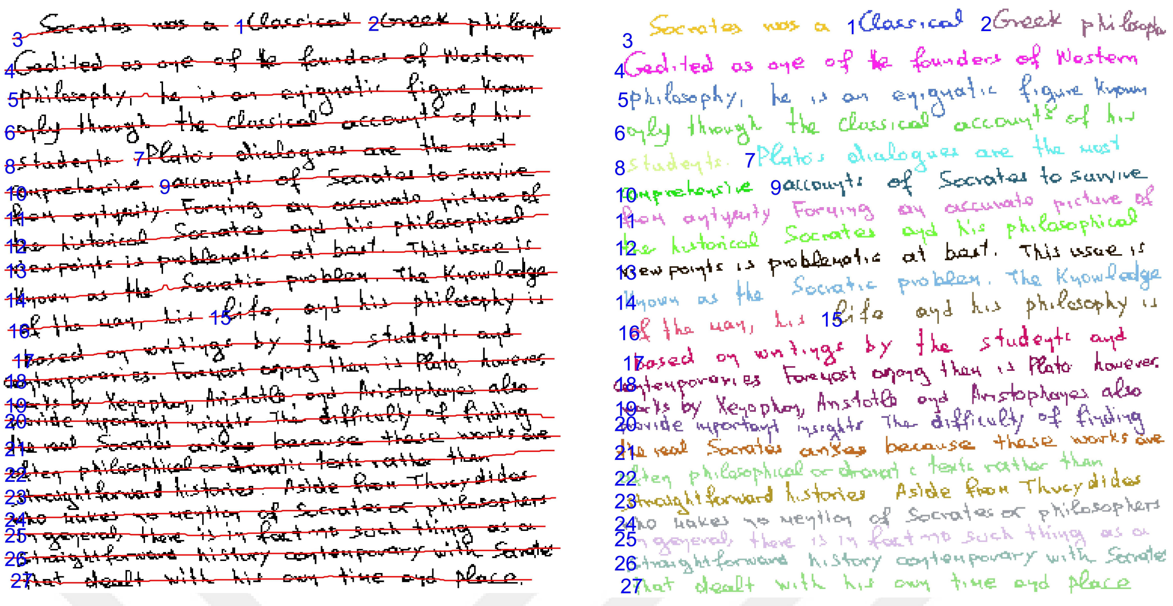
Optimum Angles Estimation

In the final divided image into 12 $\times$ 8 parts, it can be said that the skew angle will remain the same in each of it. There can be a slight difference in inclination of text lines within one divided part, but it can be compromised as the difference cannot that much, which may affect the final results.

The angles in each of 12 $\times$ 8 parts are calculated by using the mean value which is calculated by Eq. 2.12:

$$std_p(k,\theta) \;=\; std(Me(k,\theta)) \tag{2.13}$$

where, k represents one of the parts from 12×8 parts, θ is the respective estimated



(a) (b)

Figure 2.17. (a) Text Lines Detected shown in red colour and (b) Text lines differentiated with different colours.

skew angle, $Me(k, \theta)$ is the mean value of rows of kth vertical part.

The skew angle for each divided part of the original image will be the one with the maximum value of std_p. The estimated skew angles are shown in Fig. 2.16a. These are the possible skew angles at their respective parts.

Final Filtered Image

A new final filtered image is generated using skew angles in Eq. 2.11, in which the text lines are detected as darker portions, with the same movement of each text line horizontally as it is in the original text, irrespective of any intervention of alphabets between two text lines. The filtered image at skew angles is also shown in Fig. 2.16b.

Accurate estimation of skew angles is a must, as it affects the final filtered image. There might be some difference in the actual and estimated skew angle in Fig. 2.16a, but still the final filtered image can trace out text lines with best high accuracy. This image can be used for text lines detection, as the alphabets overlapping between the text lines is diminished.

2.2.3. Text Lines Detection

Fig. 2.16b shows the text lines that does not have any overlapping between the text lines. A peak finding algorithm is applied in vertical direction column by column on Fig. 2.16b. Peak Ps in column x can be calculated by:

$$Ps(x : x + L) \;=\; \begin{cases} 1, & \text{if } \mathrm{mean}(Im(x : x + L, :)) \;\geq\; Lim \\ 0, & \text{if } \mathrm{mean}(Im(x : x + L, :)) \;<\; Lim \end{cases} \tag{2.14}$$

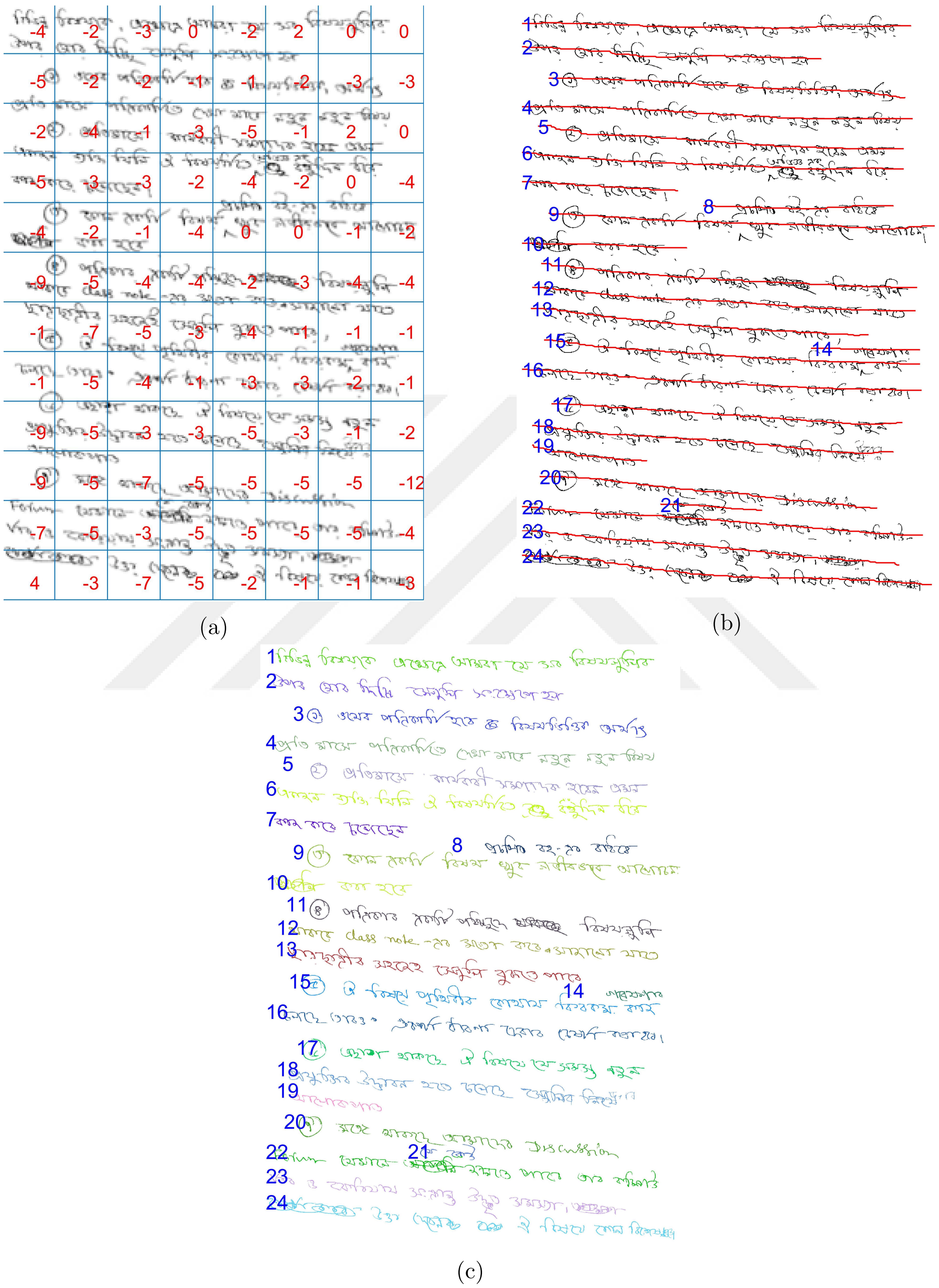
where $Lim \;=\; \mathrm{mean}(Im(x : x + L, :)) - \min(Im(x : x + L, :))$ and L represents set of columns where peaks are determined by averaging all these columns before finding the peaks.

Text Lines detected using Eq 2.14 are shown in Fig. 2.17. The text lines in this document are much closer to each other, still, they are detected correctly. There are some text lines which are divided into two or more parts, because of an increase in inter words spacing, i.e., text line numbers $1, 2, 3$. This division of text lines is not counted as an error, because it will not effect the final alphabets detection or the sequence of the detected alphabets of the document.

The algorithm presented here can also be useful in almost all languages, as all languages are written in the same format. Fig. 2.18, Fig. 2.19 and Fig. 2.20 shows results of Bangla [16, 17], English [34] and Arabic [34] languages.

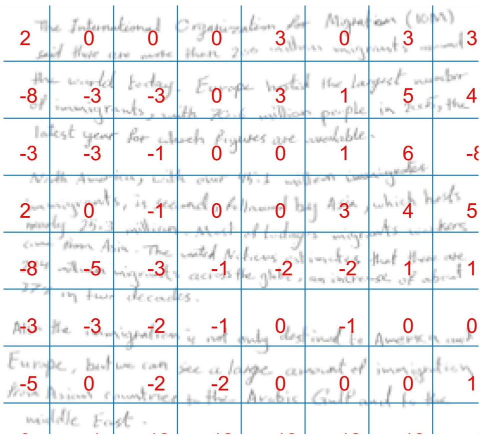
The text lines detected in Fig. 2.18 are inclined in the downward direction. Some of the text lines are starting from the mid of paper i.e., text line number $8, 14$ and 21. The algorithm has detected even a small text line number 21 which is small in length and is also surrounded by the text lines from upward and downward direction. The text lines spacing is also much smaller than the rest of the document. The proposed algorithm has accurately detected text lines of such a complex document.

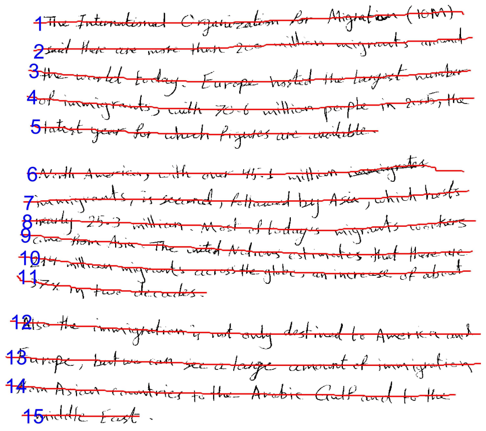
Fig. 2.19 shows a much simpler document compared to the earlier one. The curvy



(a) (b) (c)

Figure 2.18. Text Line detection applied on Bangla language.


(a)


(b)

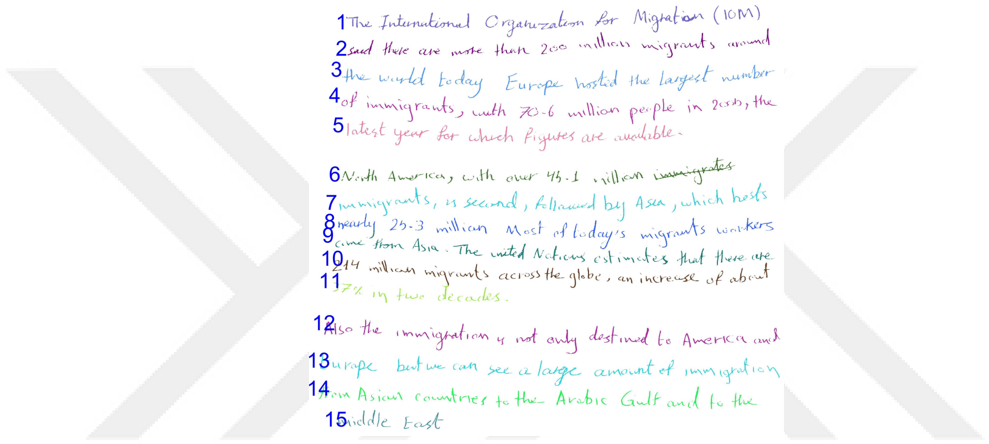
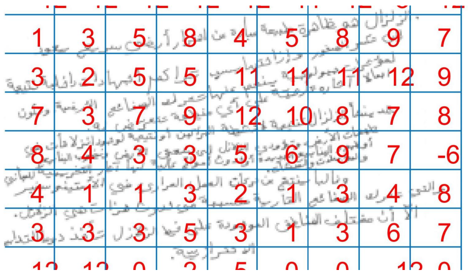
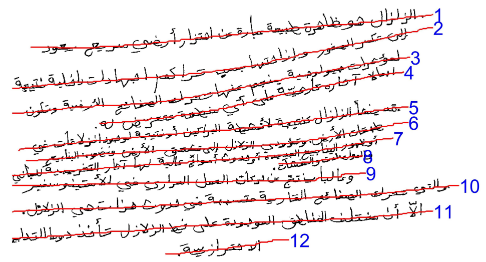

(c)

Figure 2.19. Text Line Detection applied on English language.


(a)


(b)

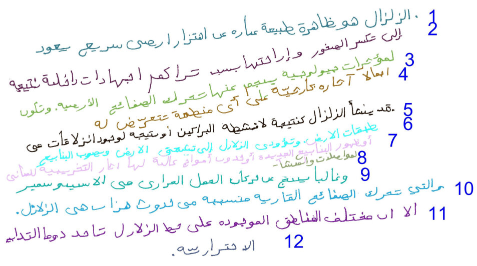

(c) .

Figure 2.20. Text Line Detection applied on Arabic language.

text lines i.e., text line number $3, 4$ are accurately detected.

Fig. 2.20 shows a script written in the Arabic language. Unlike Bangla and English languages Arabic is written from right to left. The text lines are much more inclined, closer to each other, also have some small in length text lines, but are detected with best high accuracy.

2.2.4. Conclusion

At every stage, the output image is masked using a mask which is constructed by averaging the original grey scale image using a square averaging function of size $20\times\ 20$ pixels. This lets us remove the unwanted blank part of the document so that it does not affect our later processes.

The detected text lines are used to find alphabets which lie under those detected text lines. The connected component technique is used to find alphabets in the text lines.

The total time for text lines tracking depends upon the size of the original image. As the size of scanned documents differ in all cases, the time for text lines detection may also vary. Table 2.2 shows the size of images which are used in this chapter for text line detection. Bigger the size of the image, more will be the time and power required for line detection.

To reduce the processing time, the original image in the pre-processing step is resized to a smaller one, i.e. $[400\quad NaN]$ in this case. Width of images is reduced to 400 pixels and height will also be decreased accordingly. Table 2.3 shows the decrease in time if the size of the original image is decreased.

The text line detection algorithm is compared with the algorithm mentioned in [28]. The proposed algorithm was tested on 200 images of IAM dataset [45]. The total number of text lines detected are 1867 out of which 7 text lines are found to be a false detection, which gives an accuracy of 99.63 % which is better than the one listed in

Table 2.1. Although the text lines in IAM dataset are almost horizontal.

Table 2.2. Time consumption for text lines detection.

	Dimensions without resizing ($w \times h$)	Time without resizing (sec)	Dimensions after resizing ($w \times h$)	Time with resizing (sec)
Fig. 2.17	393 × 371	15.54	400 × 378	16.042
Fig. 2.18	2263 × 1572	222.40	400 × 278	13.52
Fig. 2.19	3359 × 2471	944.40	400 × 295	19.12
Fig. 2.20	3347 × 2471	843.54	400 × 296	17.69

Table 2.3. Decrease in size and time.

	Decrease in Dimension	Decrease in Time
Fig. 2.17	−3.70 %	−3.22 %
Fig. 2.18	96.87 %	93.92 %
Fig. 2.19	98.58 %	97.98 %
Fig. 2.20	98.56 %	97.90 %

2.3. Text Character Recognition

The field of handwritten character recognition is very important for offline recognition. As offline character recognition is more difficult than online character recognition systems [18]. The online characters also have the information of sequence or direction in which they are written. The ability to deal with a large amount of data in a certain context will bring a lot of value. One example of these applications is to automate the text transcription process that intended to be applied on the ancient documents and considering its complexity and irregularity nature due to of the manual aspects of writing [52].

Deep Learning (DL) is the new application of machine learning for learning representation of data. Deep Learning architecture had also won the ImageNet classification challenge in 2012 [53]. Since that time, Deep Learning based architectures had won many challenges and are still in progress.

2.3.1. Convolutional Neural Network

Convolutional neural networks are currently one of the most prominent algorithms for deep learning with image data. For traditional machine learning, relevant features have to be extracted manually whereas, deep learning uses raw images as input to learn certain features. Convolutional Neural Network (CNN) consists of an input and output layer, and several hidden layers between the input and output. Examples of in between layers are convolutional layers, max-pooling layers and fully connected layers [54].

CNN architectures vary in the number and type of layers implemented for its specific application. For continuous responses, the network should include a regression layer at the end of a network, whereas for categorical responses the system must include a classification function and layer. Neurons in each CNN layer are arranged in a 3D arrangement and transform a three-dimensional output from a three-dimensional input. For our particular application, the input layer holds the images as 3D inputs, with the height, width and RGB values as dimensions. Hereafter, in the convolutional layer neurons are attached to the regions of the image and transformed into a three-dimensional output [54]. An example of a simple network [55] is also shown in Fig. 2.21.

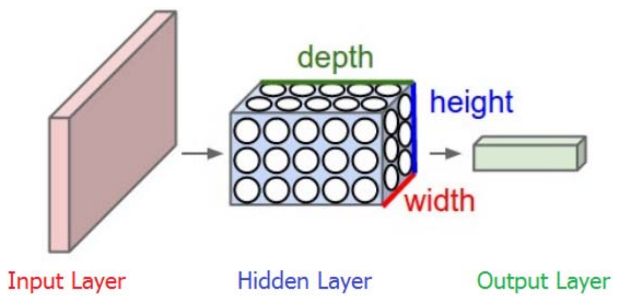


Figure 2.21. A basic CNN where the red input layer shows input image is transformed into a 3D arrangement.

CNN configurations comprise a multitude of hidden layers. In each layer, activation volumes are altered with the use of differentiable functions. Four principle layer types exist that are used to build CNN configurations, an example [56] is illustrated in Fig. 2.22.

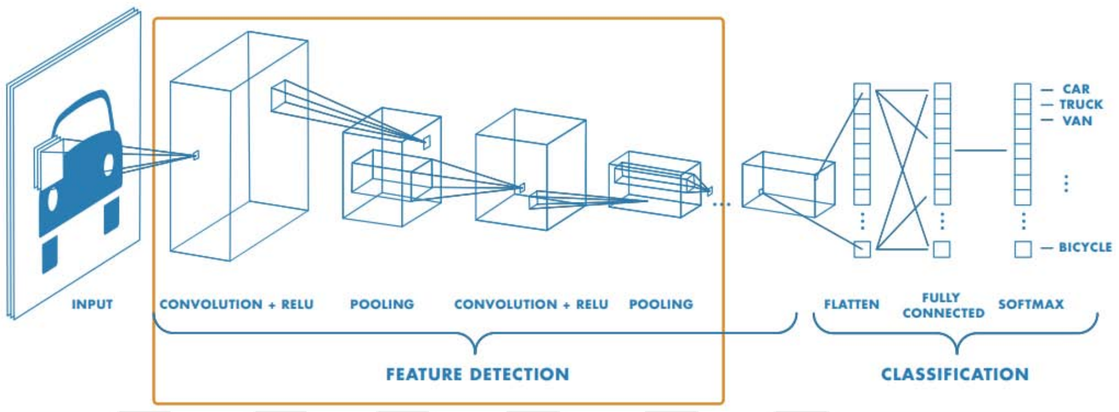


Figure 2.22. Convolutional neural network architecture that classifies input images as belonging to a number of categories including cars, trucks, vans and bicycles.

1. *imageinputLayer*: An image input layer inputs 2-D images to a network and applies data normalization [57].

2. *convolution2dLayer*: A 2-D convolutional layer applies sliding convolutional filters to the input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input and then adding a bias term [58].

3. *leakyReluLayer*: A leaky ReLU layer performs a threshold operation, where any input value less than zero is multiplied by a fixed scalar [59].

4. *batchNormalizationLayer*: A batch normalization layer normalizes each input channel across a mini-batch. To speed up the training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers [60].

5. *dropoutLayer*: A dropout layer randomly sets input elements to zero with a given

probability. Dropout layer is used to prevent the neural network from overfitting [57, 61].

6. *maxPooling2dLayer*: A max pooling layer performs down-sampling by dividing the input into rectangular pooling regions, and computing the maximum of each region [62].

7. *additionLayer*: An addition layer adds inputs from multiple neural network layers element-wise.

8. *fullyConnectedLayer*: A fully connected layer multiplies the input by a weight matrix and then adds a bias vector. Understanding the difficulty of training deep feedforward neural networks

9. *softmaxLayer*: A softmax layer applies a softmax function to the input [63].

10. *classificationLayer*: A classification layer computes the cross-entropy loss for multi-class classification problems with mutually exclusive classes. The layer infers the number of classes from the output size of the previous layer [63].

2.3.2. Building a CNN in MATLAB

The framework described in this section is developed in MATLAB. Two MATLAB toolboxes are used: The Parallel Computing and Deep Learning Toolboxes. The parallel toolbox is used to speed up the task by using GPUs of a desktop computer. Deep Learning Toolbox allows to build and edit deep learning networks interactively using the Deep Network Designer app. This app can be used to:

i. Import and edit networks.

ii. Build new networks from scratch.

iii. Drag and drop to add new layers and create new connections.

iv. View and edit layer properties.

v. Generate MATLAB code.

A basic 16 layer neural network mentioned in [56], used for handwritten digits recognition, is used here at the initial stage for alphabets recognition. More advanced versions of this network are trained for alphabets recognition by increasing the numbers of layers, changing the filter sizes of layers, changing the layers itself and changing different training parameters. More than 30 different neural networks are trained and tested with different parameters. A 49 layer network shown in Fig. 2.23 is finalized after training it for a couple of times and making sure of its best performance.

Time for training the deep convolution network shown in Fig. 2.23, may vary depending upon the size of the dataset and available processing power. There are three options available for training of a deep convolutional network:

1. CPU based computation

2. GPU based computation

3. Cloud based GPU computation

The most reliable and simpler option for training phase is CPU based computations. However, the time consumption using CPU is very high as a computer computes the task in a serial configuration. The use of Graphical Processing unit, without any further programming, can cut down the training time of CNN significantly. A CUDA based NVidea GPU is necessary with at least 3.0 to compute capability for parallel computation. Lastly, cloud based GPU computation considers the employment of cloud resources for the processing power [54]. In this research, both CPU and GPUs are used for the training of the network.

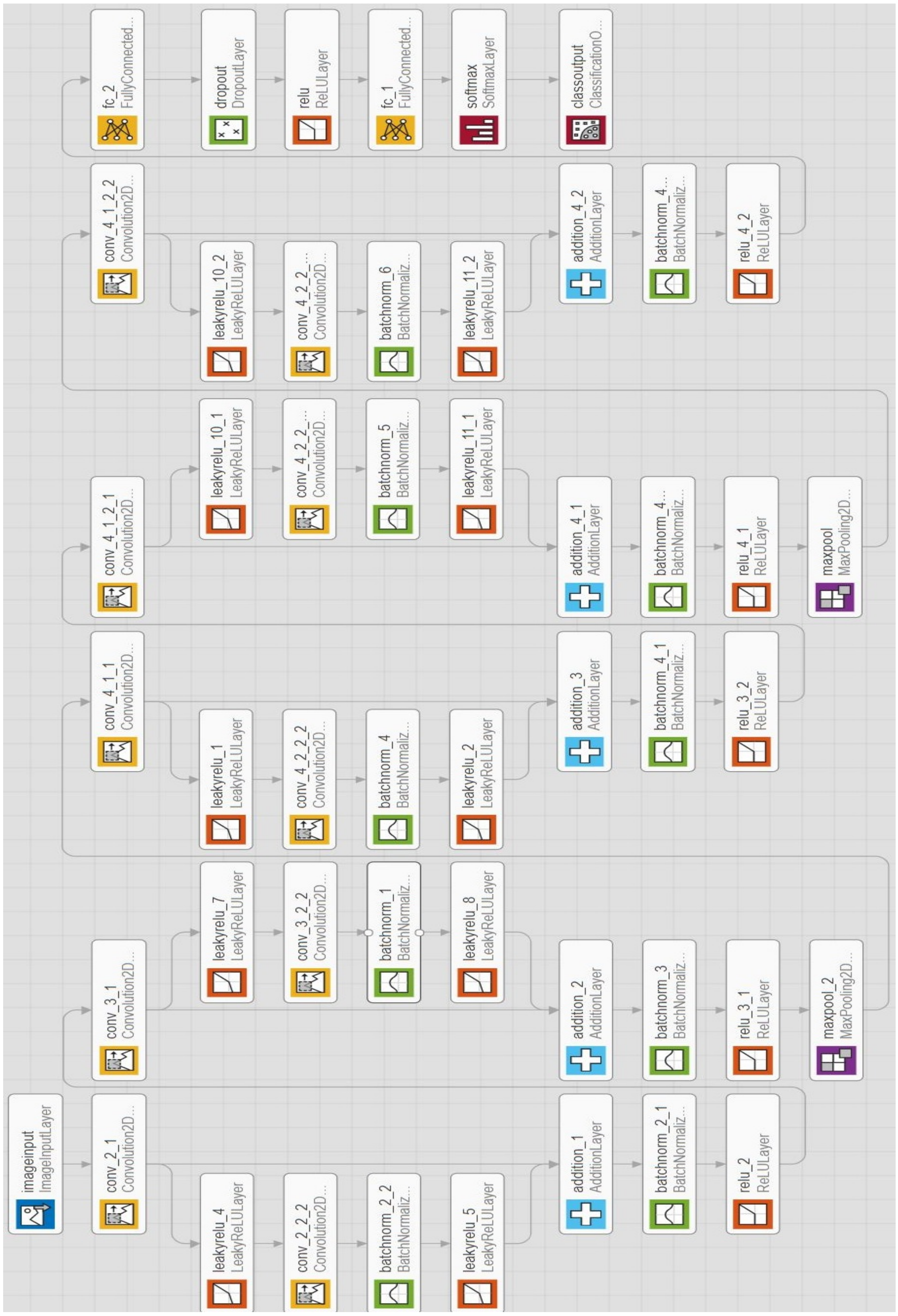
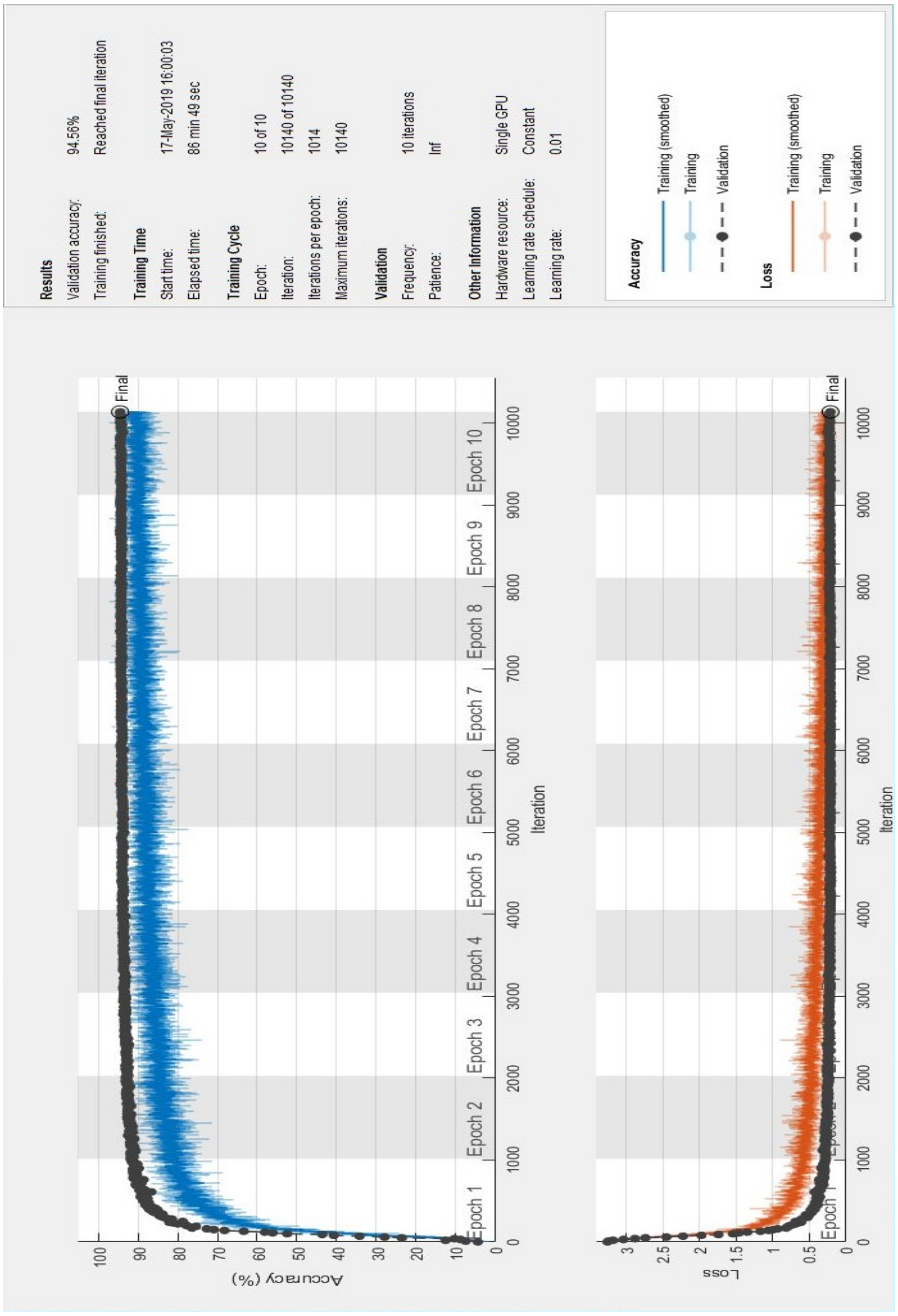

imageinput ImageInputLayer
conv_2_1 Convolution2D...
leakyrelu_4 LeakyReLULayer
conv_2_2_2 Convolution2D...
batchnorm_2_2 BatchNormaliz...
leakyrelu_5 LeakyReLULayer
addition_1 AdditionLayer
batchnorm_2_1 BatchNormaliz...
relu_2 ReLULayer
conv_3_1 Convolution2D...
leakyrelu_7 LeakyReLULayer
conv_3_2_2 Convolution2D...
batchnorm_1 BatchNormaliz...
leakyrelu_8 LeakyReLULayer
addition_2 AdditionLayer
batchnorm_3 BatchNormaliz...
relu_3_1 ReLULayer
maxpool_2 MaxPooling2D...
conv_4_1_1 Convolution2D...
leakyrelu_1 LeakyReLULayer
conv_4_2_2_2 Convolution2D...
batchnorm_4 BatchNormaliz...
leakyrelu_2 LeakyReLULayer
addition_3 AdditionLayer
batchnorm_4_1 BatchNormaliz...
relu_3_2 ReLULayer
conv_4_1_2_1 Convolution2D...
leakyrelu_10_1 LeakyReLULayer
conv_4_2_2_... Convolution2D...
batchnorm_5 BatchNormaliz...
leakyrelu_11_1 LeakyReLULayer
addition_4_1 AdditionLayer
batchnorm_4... BatchNormaliz...
relu_4_1 ReLULayer
maxpool MaxPooling2D...
conv_4_1_2_2 Convolution2D...
leakyrelu_10_2 LeakyReLULayer
conv_4_2_2_... Convolution2D...
batchnorm_6 BatchNormaliz...
leakyrelu_11_2 LeakyReLULayer
addition_4_2 AdditionLayer
batchnorm_4... BatchNormaliz...
relu_4_2 ReLULayer
fc_2 FullyConnected...
dropout DropoutLayer
relu ReLULayer
fc_1 FullyConnected...
softmax SoftmaxLayer
classoutput ClassificationO...

Figure 2.23. 49 layer Deep Neural Network with varying filter sizes, used for handwritten character recognition developed in MATLAB using Deep Network Designer.



Results

Validation accuracy: 94.56%
Training finished: Reached final iteration

Training Time

Start time: 17-May-2019 16:00:03
Elapsed time: 86 min 49 sec

Training Cycle

Epoch: 10 of 10
Iteration: 10140 of 10140
Iterations per epoch: 1014
Maximum iterations: 10140

Validation

Frequency: 10 iterations
Patience: Inf

Other Information

Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.01

Figure 2.24. Results of training on MATLAB.

2.3.3. CNN Training Phase

As discussed in Section 2.1, a combination of EMNIST dataset [20] and self-prepared dataset is used for the training phase. The total number of images are $202,800$, which includes 7800 images per alphabet. The size of each image is 28×28 pixels and is saved in binary format. All the data is labelled from 1 to 26 representing each alphabet by placing in a folder named from 1 to 26. Fig. 2.26 shows some examples of alphabets taken from the combined dataset.

The training variables and the values used are listed below:

1. $MaxEpochs = 10$,
2. $ValidationFrequency = 10$,
3. $MiniBatchSize = 150$,
4. $TrainingData = 202,800 \times 0.75 = 152,100$ and
5. $ValidationData = 202,800 \times 0.75 = 50,700$.

One Epoch is passing an entire dataset forward and backwards through the neural network during the training phase to update the values of neurons. Since passing a dataset at once to the network is difficult, so the dataset is divided into small batches. Batch size represents, how many images are passed to the network at once. One epoch leads to underfitting of the curve in the graph. As the number of epochs increases, more the number of times the weight are changed in the neural network and the curve goes from underfitting to optimal to overfitting curve. [64].

If the training dataset size is $152,100$ and batch size is 150, one epoch will have $152,100\ /\ 150 = 1014$ iterations. As max epochs is 10, the total number of iterations in 10 epochs will be $1014 \times 10 = 10140$. Validation frequency is the number of iterations between evaluations of validation metrics.

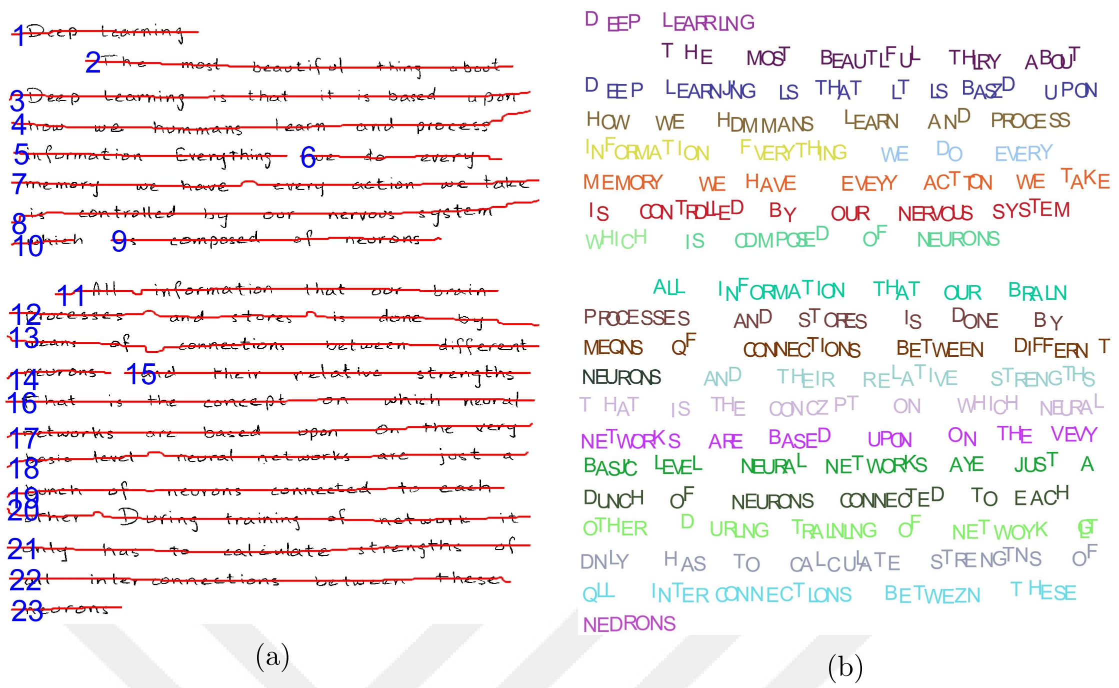


Figure 2.25. Example of my own handwriting (a) Text lines detected and (b) Predicted text.

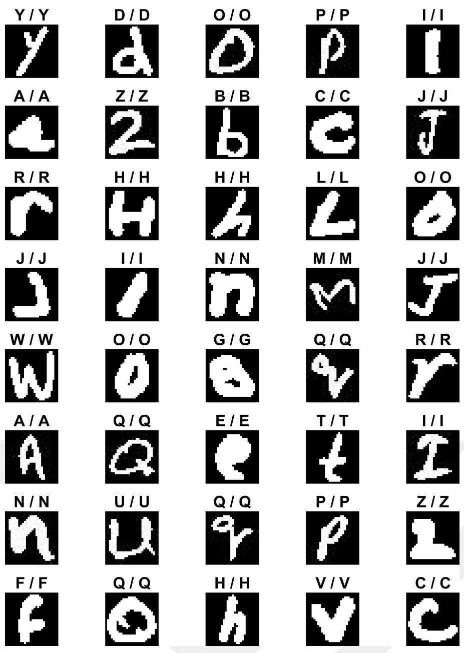
2.3.4. Conclusion

Vaidya et al. [35] discussed a handwritten character recognition method using convolutional neural network using the EMNIST dataset [20]. The accuracy of 94 % was obtained after training the neural network.

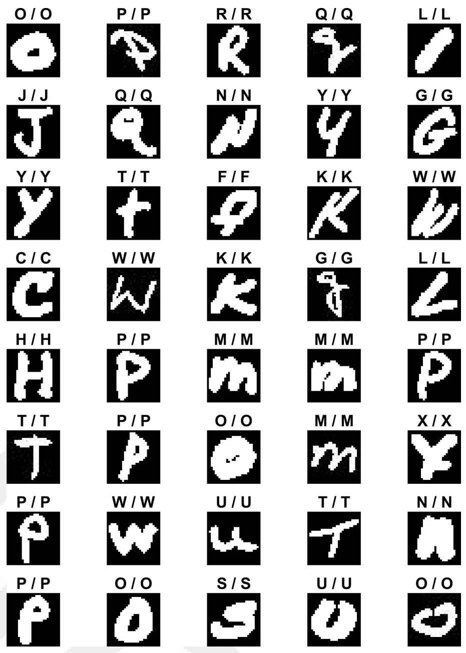
The validation accuracy for the training of the proposed neural network shown in Fig. 2.23 is 94.56 %, which gives higher accuracy compared to the one carried out by Vaidya et al. [35]. The training was carried out on the GPU system in 87 minutes. The accuracy plot is shown in Fig. 2.24.

Fig. 2.26 shows some of the predicted alphabets taken randomly from the mixed dataset which were tested using the trained network in a batch size of 40 images. The alphabets shown here were taken in a group of 40 images at one time randomly and the accuracy was calculated based on the predicted alphabets. False detections are highlighted in red colour.

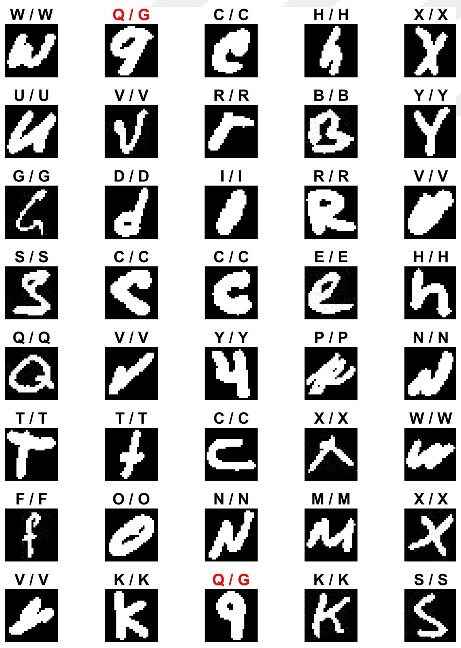
Fig. 2.25 shows a sample handwritten text document written by myself.



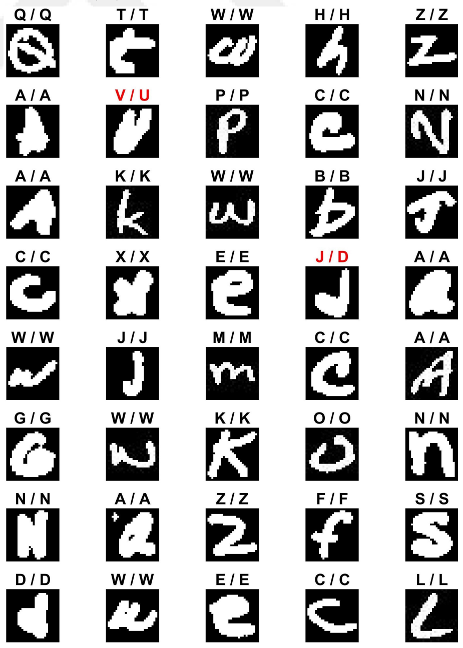
(a) Accuracy = 100 %



(b) Accuracy = 100 %



(c) Accuracy = 95 %



(d) Accuracy = 95 %

Figure 2.26. Examples of alphabets taken from the combination of EMNIST dataset and self-prepared dataset. Original and Predicted alphabets are shown on the top of each alphabet respectively. Wrong detections are shown in red colour.

3. RESULTS AND DISCUSSION

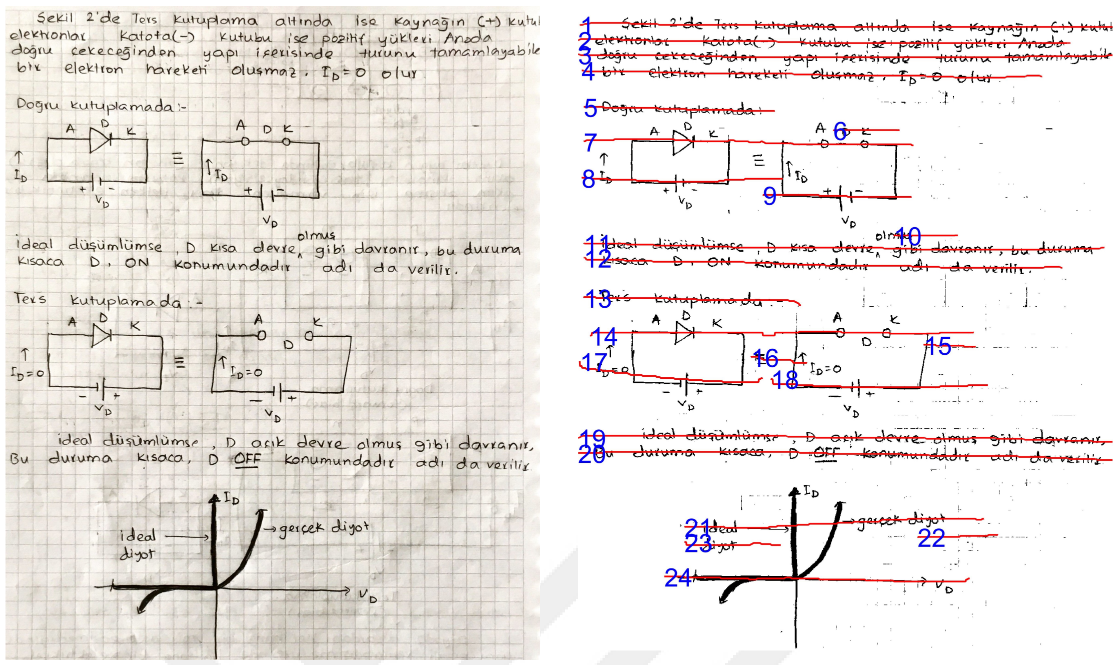
This thesis discusses the method of text-lines detection, alphabet detection and alphabet recognition in Chapter 2. After successful detection of text lines, the next phase is alphabets detection. At this stage the ideal case is used i.e., the alphabets in a text lines are not connected to each other, which is not the case in reality. The alphabets may interfere with each other within a text line. Some of the examples are also discussed in Chapter. 2.

The current research work focuses on the text document which does not have any of the graphical objects in it. But in the real case, the text documents, mostly the ones prepared by students during their lectures do contain graphical objects like figures, plots or sketches. In the case of such documents, the first step is to detect and remove graphical objects out of the scanned documents.

3.1. Graphics Extraction

The proposed algorithm can also be used in classifying graphics from a scanned document. One of the scanned document which contains some graphics is also shown in Fig. 3.1a. Text lines detection algorithm is applied on this image and the result is shown in Fig. 3.1b. The text lines are detected correctly beside, there is some detection of non-textual objects in it.

The graphical objects can also be differentiated while dividing the original image into parts. Fig. 3.2a shows scanned document where the original image was divided and the objects which lie on the boundaries are labelled as false objects and are marked in red colour here. The False objects are then scrutinized such that the group of false objects which are connected with each other and the sizes are more than the average of all the objects, are classified as graphical objects. The graphical objects are shown in Fig. 3.2b.



Şekil 2'de Ters kutuplama altında ise kaynağın (+) kutul elektronlar katota(−) kutubu ise pozitif yükleri Anoda doğru çekeceğinden yapı içerisinde durumu tamamlayabilir bir elektron hareketi oluşmaz. $I_D = 0$ olur

Doğru kutuplamada:-

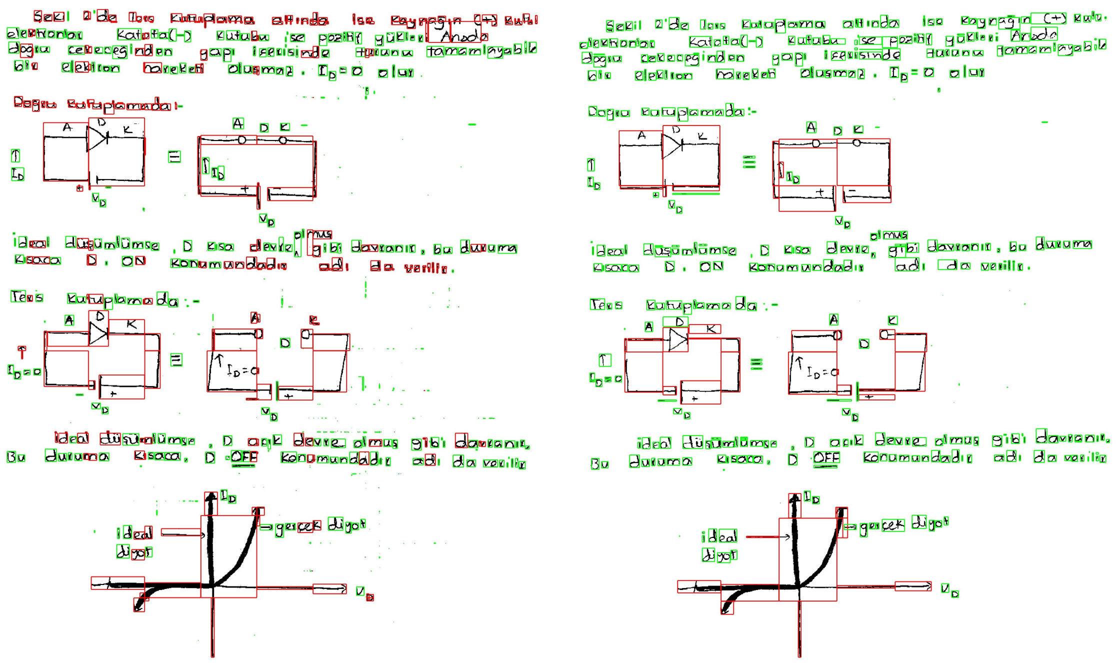
ideal düşümlümse, D kısa devre olmuş gibi davranır, bu duruma kısaca D, ON konumundadır adı da verilir.

Ters kutuplamada:-

ideal düşümlümse, D açık devre olmuş gibi davranır, Bu duruma kısaca, D OFF konumundadır adı da verilir.

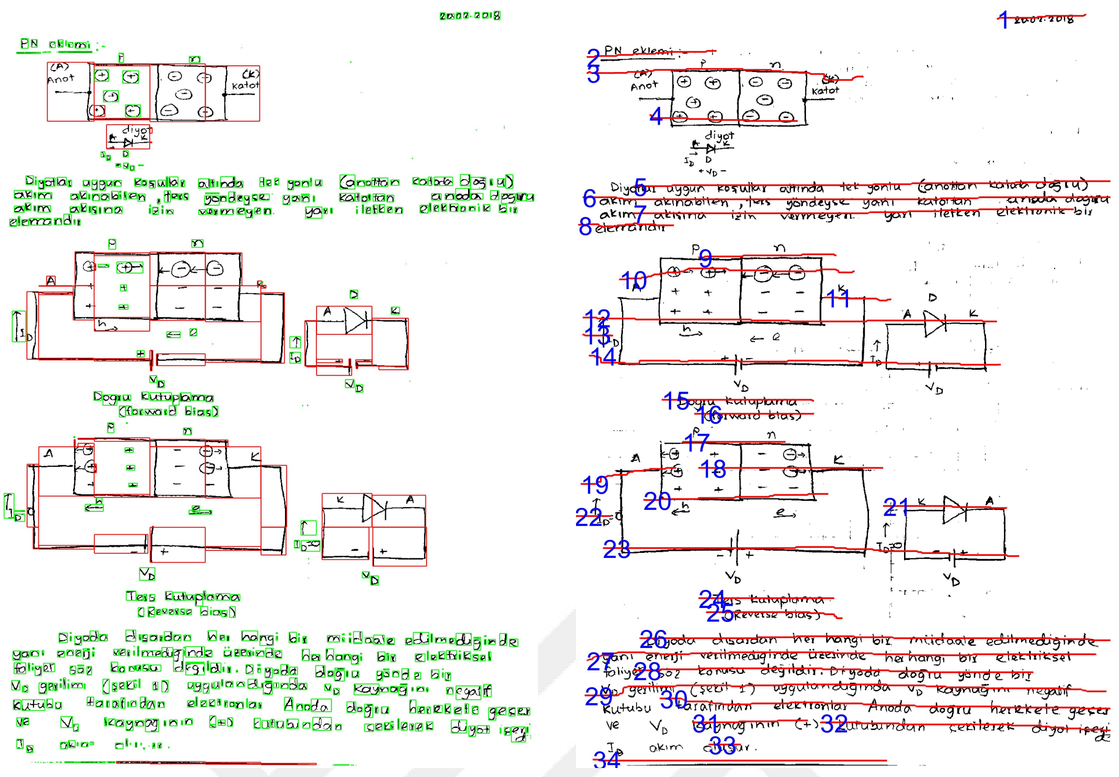
(a) (b)

Figure 3.1. (a) Original image with graphical objects and (b) Text Lines algorithm applied.



(a) (b)

Figure 3.2. (a) False objects detected and (b) Graphical objects detected.



(a) (b)

Figure 3.3. (a) Graphical Objects Detection and (b) Text Line Detections.

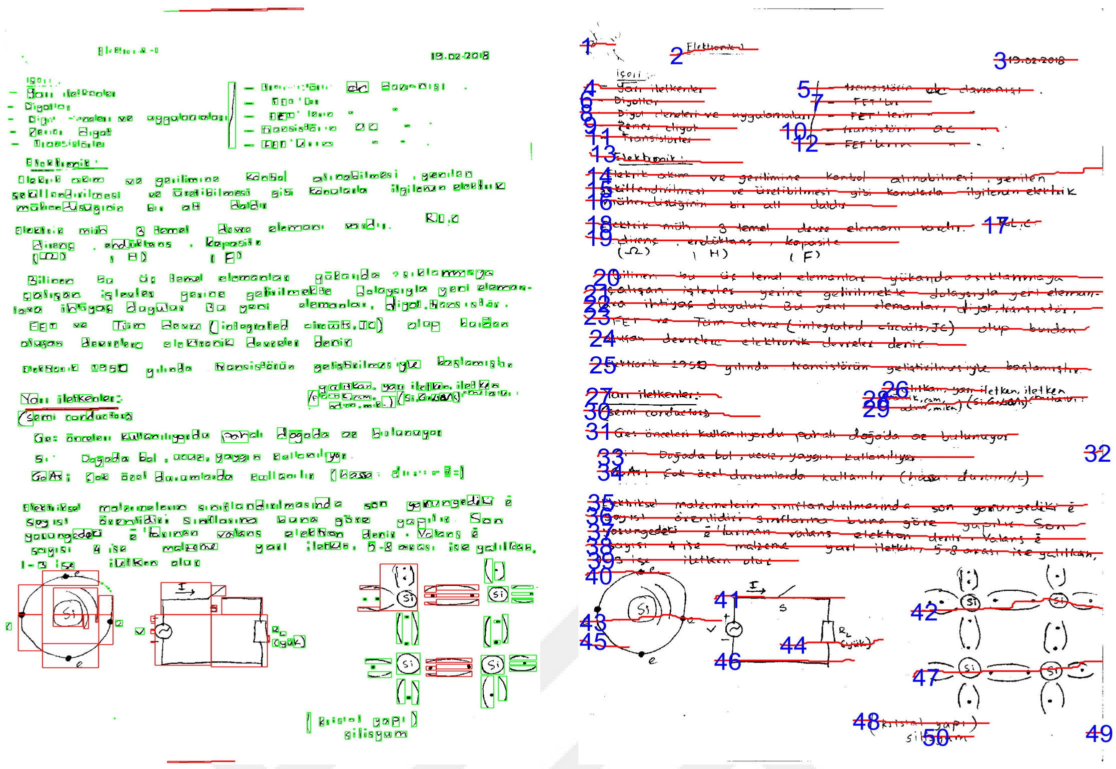
Fig. 3.3 & Fig. 3.4 shows two other examples of handwritten text documents which contain graphs and figures in it. Fig. 3.3a & Fig. 3.4a shows figures extracted by dividing the original image into parts. The noticing thing is the output that gives very good result in excluding those figures.

Fig. 3.3b & Fig. 3.4b show text lines detection algorithm results. The text lines are detected very efficiently even with the presence of graphical objects.

By comparing the results of Fig. 3.3 & Fig. 3.4, combining both outputs, i.e., graphical objects extraction and text line detection can give better results in such types of documents.

3.2. MATLAB GUI

A GUI is developed to implement all the research work explained in previous the chapter using MATLAB application designer. GUI has the option of changing some



(a) (b)

Figure 3.4. (a) Graphical Objects Detection and (b) Text Line Detections.

variables depending upon the input document. The algorithms discussed in Chapter 2 are implemented with a user-friendly interface giving a variety of options to the user, starting from uploading input image to getting predicted text.

3.2.1. How To Use

The GUI consists of two main panels, each has own axes called input and output side. Each panel has options, to show any of the intermediate results, under each axes. These two panels can be used to compare any of the two intermediate results. Fig. 3.5 shows the GUI developed for this research work. The input side is showing text lines detected, whereas, the output panel is showing the predicted text extracted from the input image.

The options to show image in any of the panels include, input greyscale image, input binary image, resized image, a filtered image at a random angle, an image with estimated skew angles, a final filtered image at the respective estimated skew angles, text lines detected, words detected and predicted text.

The information of GUI shown in Fig. 3.5 is listed below:

1) represents number of columns represented by Lim in Eq. 2.14 for finding peaks. Default value of Lim is 8,

2) represents the height of averaging filter at different angles represented by n in Eq. 2.11. The default value of n is 3,

3) represents the number of horizontal divisions of the original image. The default horizontal divisions are 8,

4) represents the height of mean filter represented by h_p in Eq. 2.12. The default value of h_p is 4,

5) represents limits of angles in Eq. 2.11 and Eq. 2.12. The default angles span consists of -12 to $+12$,

6) represents the number of vertical divisions of the original image. The default vertical divisions consist of 12 parts,

7) represents the type of pre-processing filter used on the original image. The default filter is Adaptive Binarization,

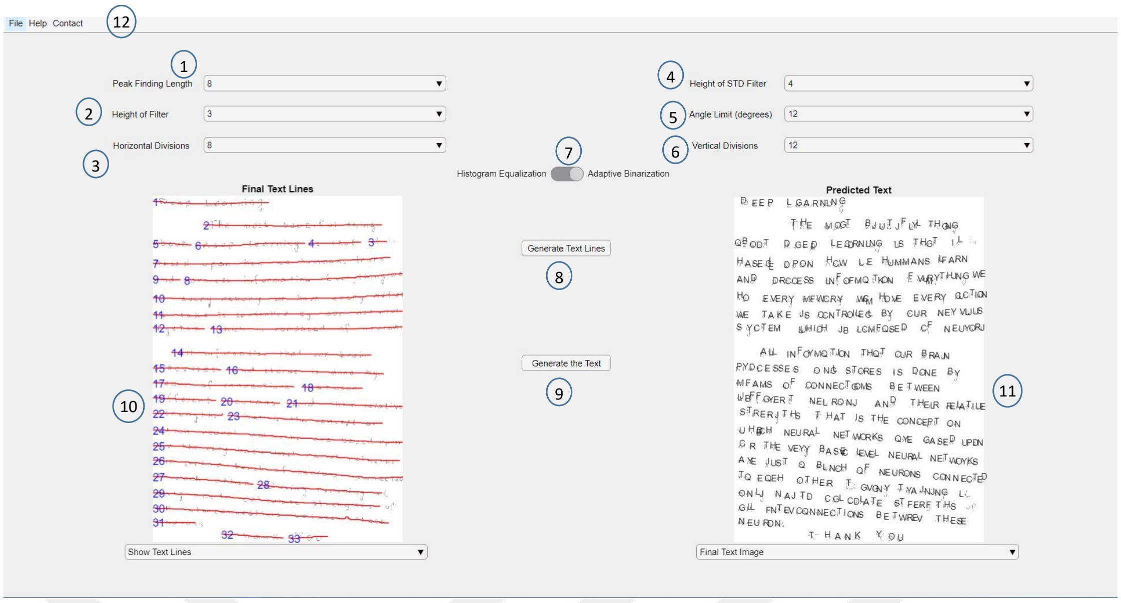
8) generates text lines of the input image,

9) generates predicted text extracted from the text lines.

10) input panel,

11) output panel and

12) menu bar to upload or save an image.



Peak Finding Length: 8
Height of Filter: 3
Horizontal Divisions: 8
Height of STD Filter: 4
Angle Limit (degrees): 12
Vertical Divisions: 12

```
DEEP LGARNING
        THE MOGT BJUTJFLYL THGNG
QBODT DGED LEGRNING LS THGT IL
HASEG DPON HCW LE HUMMANS LFARN
AND DRCCESS INFOFMQTKON EVURYTHING WE
HO EVERY MEWCRY WGM HOVE EVERY QCTION
WE TAKE JS CCNTROLLEG BY CUR NEYVUJUS
SYCTEM WHICH JB LCMFQSED CF NEUYORJ
    ALL INFOYMQTJON THQT CUR BRAJN
PYDCESSES ONG STORES IS DONE BY
MFAMS OF CONNECTGOMS BETWEEN
UBFFGYERT NELRONJ AND THELR FELATILE
STRERJTHS THAT IS THE CONCEPT ON
UHBCH NEURAL NETWORKS QYE GASED UPON
GR THE VEYY BASE LEVEL NEURAL NETWOYKS
AYE JUST Q BLNCH QF NEURONS CONNECTED
TQ EQEH OTHER T GVGNY TYAJNJNG L
ONLJ NAJTD CGL CDLATE STFERFTHS
GLL FNTEVCONNECTIONS BETWREV THESE
NEURON
        THANK YOU
```

Figure 3.5. GUI with its options numbered and explained.

The process starts with uploading the image from the menu bar (12). The greyscale image will be automatically shown in the input panel of the GUI. The default pre-processing technique will be applied on the input image before showing it in the input panel. The user has to check the results of the pre-processing technique and can change from the toggle button at number 7. Switching the toggle to the other side will automatically apply the respective filter on the image and will show the image in the input panel.

After pre-processing, the user has to click on Generate Text Lines (8) to start the algorithm of text lines detection. At the end of text lines detection algorithm, the final text lines will be automatically shown at the output panel.

The user can check the final detection of text lines from the output panel. If the results are not up to the mark, the user can change the parameters (1 - 6) in Fig. 3.5 and run the algorithm again.

After the successful detection of text lines, the user has to click on Generate the Text (9) to start the algorithm, which uses a pre-trained neural network to predict the alphabets detected in the last step. The recognition of characters starts with the alphabets detection technique from the text lines detected.

4. CONCLUSION AND FUTURE WORK

The research work carried out consists of three phases. Preparation of handwritten dataset, detection of text lines and handwritten characters recognition discussed in Chapter 2. The dataset here is prepared to increase the accuracy of own handwriting. The text lines detection algorithm successfully detects all kinds of text lines from almost all kind of languages with best high accuracy. Text recognition carried out using the neural network is best high accuracy.

Within the scope of this research work, the text documents used for text lines detection does not contain any of the graphical objects in it. But in a real case, most of the documents especially the one prepared by the students does contain graphical objects in it. As discussed in Chapter 3 while dividing the original image, false objects can be used as a base for the improvement in accuracy of graphical objects extraction.

In this research work although the text lines are detected accurately at any angle, finding correct alphabets from those text lines is another complete long term research work. Here the ideal case is used, where the alphabets are not connected to each other in a text line, which is not the case in reality.

The current output which GUI can give is shown in Fig. 3.5, where all the alphabets are placed at the exact location where those alphabets are found in the original image. This makes the output little difficult to read.

An edit box can be added in the GUI where all the detected alphabets can be shown with an edit option, where a user can edit any of the alphabets in that edit box if it is a wrong detection. The wrong detections are extracted from the original image and placed in a separate folder with their correct labels. These alphabets can be used to train the network using Transfer Learning to improve the accuracy of the network in the future.

5. REFERENCES

1. Barnard, N., The origins of Chinese civilization, volume 1. Univ of California Press, 1983.
2. Bright, W., The blackwell encyclopedia of writing systems by florian coulmas. Language, 74, 1 (1998) 206–206.
3. Haarmann, H., Geschichte der Schrift, CH Beck, 2198 (2002).
4. Cortada, J. W., Before the computer: IBM, NCR, Burroughs, and Remington Rand and the industry they created. Princeton University Press, 2000, 1865-1956.
5. Richards, G. T., The history and development of typewriters. HM Stationary Office, 1964.
6. Schantz H. F., History of OCR, optical character recognition. Recognition Technologies Users Association, 1982.
7. Legg, S. and Hutter M., A collection of definitions of intelligence. Frontiers in Artificial Intelligence and applications, 157, 17 (2007).
8. Kaplan, A. and Haenlein, M., Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. Business Horizons, 62 ,1 (2019) 15–25.
9. Harvey C., Artificial intelligence vs Machine Learning vs Deep Learning. www.datamation.com/artificial-intelligence/ai-vs-machine-learning-vs-deep-learning.html 7 May 2018.
10. Russell, S. J. and Norvig, P., Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, 2016.
11. Shwartz, S. S. and Ben-David, S., Understanding machine learning: From theory to algorithms. Cambridge university press, 2014.
12. Bilal A., Artificial neural networks and deep learning, www.becominghuman .ai/artificial-neural-networks-and-deep-learning-a3c9136f2137 31 Jan 2018.
13. Nasrabadi, N. M., Pattern recognition and machine learning. Journal of Electronic Imaging, 16 ,4 (2007) 049901.
14. Torrey, L. and Jude Shavlik, J., Transfer learning. Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, 1, 242 (2009).
15. Niklas Donges N., Transfer learning, www.towardsdatascience.com/transfer-learning- 946518f95666 31 Dec 2018.

16. Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M. and Basu, D. K., Cmaterdb1: a database of unconstrained handwritten bangla and bangla–english mixed script document image. International Journal on Document Analysis and Recognition (IJDAR), 15, 1 (2012) 71–83.

17. Basu, S., Chaudhuri, C., Kundu, M., Nasipuri, M. and Basu, D. K., Text line extraction from multi-skewed handwritten documents. Pattern Recognition, 40, 6 (2007) 1825–1839.

18. Plamondon, R. and Srihari, S. N., Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22, 1 (2000) 63–84.

19. Arica, N. and Yarman-Vural, F. T., An overview of character recognition focused on off-line handwriting. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 31, 2 (2001) 216–233.

20. Cohen, G., Afshar, S., Tapson, J. and Schaik, A., Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373, 2017.

21. Grother, P. J., NIST special database 19. Handprinted forms and characters database, National Institute of Standards and Technology, 1995.

22. Kurzweil, R., Richter, R., Kurzweil, R. and Schneider, M. L., The age of intelligent machines, MIT press Cambridge, 579 (1990).

23. Michalski, R. S., Carbonell, J. G. and Mitchell, T. M., Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.

24. Bhattacharya, U. and Chaudhuri B. B., Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31, 3 (2009) 444–457.

25. Pal, U., Sharma, N.,Wakabayashi, T. and Kimura, F., Handwritten numeral recognition of six popular indian scripts. Ninth IEEE International Conference on Document Analysis and Recognition (ICDAR), 2 (2007) 749–753.

26. Allum, D. R., Johns, F. S. and Clysdale, D. G., Mail encoding and processing system, May 30 1995.

27. Kurzweil, R., Kurzweil computer products. www.kurzweiltech.com/kcp.html 1974.

28. Santos, R. P., Clemente, G. S., Ren, T. I. and George,D. C., Cavalcanti. Text line segmentation based on morphology and histogram projection. 10th IEEE International Conference on Document Analysis and Recognition (ICDAR) (2009) 651–655.

29. Shyam, D., Wang, Y. and Kot, A. C., Histogram-based fast text paragraph image detection. IEEE Symposium Series on Computational Intelligence (2015) 473–480.

30. Singh, B. M., Chand, V., Mittal, A. and Ghosh, D., Text line extraction from complex layout documents. International Journal of Computer Applications, 975, 8887 (2011).

31. Rajput, G. G., Ummapure, S. B. and Patil, P. N., Text-line extraction from handwritten document images using histogram and connected component analysis. International Journal of Computer Applications, 975, 8887 (2015).

32. Barlas, P., Adam, S., Chatelain, C. and Paquet, T., A typed and handwritten text block segmentation system for heterogeneous and complex documents. 11th IEEE International Workshop on Document Analysis Systems (IAPR) (2014) 46–50.

33. Yao, C., Bai, X. and Liu, W., A unified framework for multioriented text detection and recognition. IEEE Transactions on Image Processing, 23, 11 (2014) 4737–4749.

34. Hassaïne, A., Al Maadeed, S., Aljaam, J. and Jaoua, A., Icdar 2013 competition on gender prediction from handwriting. 12th IEEE International Conference on Document Analysis and Recognition (ICDAR) (2013) 1417–1421.

35. Vaidya, R., Trivedi, D., Satra, S. and Pimpale, M., Handwritten character recognition using deep-learning. Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (2018) 772–775.

36. Lu, W., Li, Z. and Shi, B., Handwritten digits recognition with neural networks and fuzzy logic. Proceedings of ICNN'95-International Conference on Neural Networks 3 (1995) 1389–1392.

37. Banumathi, P. and Nasira, G. M., Handwritten tamil character recognition using artificial neural networks. International Conference on Process Automation, Control and Computing (2011) 1–5.

38. Murthy, B. V. S., Handwriting recognition using supervised neural networks. IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339) 4 (1999) 2899–2902.

39. Pradeep, J., Srinivasan, E. and Himavathi, S., Neural network based handwritten character recognition system without feature extraction. International Conference on Computer, Communication and Electrical Technology (ICCCET) (2011) 40–44.

40. Likforman-Sulem, L., Zahour, A. and Taconet, B., Text line segmentation of historical documents: a survey. International Journal of Document Analysis and Recognition (IJDAR), 9, 2-4 (2007) 123–138.

41. O'Gorman, L., The document spectrum for page layout analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15, 11 (1993) 1162–1173.

42. Kise, K., Sato, A. and Iwata, M., Segmentation of page images using the area voronoi diagram. Computer Vision and Image Understanding, 70, 3 (1998) 370–382.

43. Jung, K., Kim, K. I. and Jain, A. K., Text information extraction in images and video: a survey. Pattern Recognition, 37, 5 (2004) 977–997.

44. Sezgin, M. and Sankur, B., Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging, 13, 1 (2004) 146–166.

45. Marti, U. V. and Bunke, H., The iam-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition, 5, 1 (2002) 39–46.

46. Bradley, D. and Roth, G., Adaptive thresholding using the integral image. Journal of Graphics Tools, 12, 2 (2007) 13–21.

47. Wellner, P. D., Adaptive thresholding for the digitaldesk. Xerox, EPC1993-110 (1993) 1–19.

48. Tcheslavski, G., Morphological image processing: Gray-scale morphology. ELEN, (2010) 4304–5365 .

49. Gatos, B., Stamatopoulos, N. and Louloudis, G., Icdar2009 handwriting segmentation contest. International Journal on Document Analysis and Recognition (IJDAR), 14, 1 (2011) 25–33.

50. Adams, R., Radial decomposition of disks and spheres. CVGIP: Graphical models and image processing, 55, 5 (1993) 325–332.

51. Simmons, G. F., Calculus with analytic geometry. McGraw-Hill New York, 1996.

52. Belaïd, A. and Ouwayed N., Segmentation of ancient arabic documents. In Guide to OCR for Arabic Scripts, Springer (2012) 103–122.

53. Sutskever, I., Hinton, G. E. and Krizhevsky, A., Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems (2012) 1097–1105.

54. Sahla, N., A deep learning prediction model for object classification.

55. Karpathy, A., Cs231n convolutional neural networks for visual recognition. Neural Networks, 1 (2016).

56. Introducing Deep Learning with MATLAB. www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/80879v00-Deep-Learning-ebook.pdf.

57. Krizhevsky, A., Sutskever, I. and Hinton, G. E., Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems (2012) 1097–1105.

58. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86, 11 (1998) 2278–2324.

59. Maas, A. L., Hannun, A. W. and Ng A. Y., Rectifier nonlinearities improve neural network acoustic models. In Proc. icml, 30 (2013) 3.

60. Ioffe, S. and Szegedy, C., Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

61. Srivastava, N., Hinton, G.,Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15, 1 (2014) 1929–1958.

62. Nagi, J., Ducatelle, F., Caro, G. A. D., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J. and Gambardella L. M., Max-pooling convolutional neural networks for vision-based hand gesture recognition. IEEE International Conference on Signal and Image Processing Applications (ICSIPA) (2011) 342–347.

63. Bishop, C. M., Pattern recognition and machine learning. springer, 2006.

64. Sharma, S., Epoch vs batch size vs iterations. www.towardsdatascience.com/epoch-vs- iterations-vs-batch-size-4dfb9c7ce9c9.

BIOGRAPHY

USAMA MUNIR was born in Pakistan. He received his bachelor degree in Electronic Engineering from Muhammad Ali Jinnah University Islamabad, Pakistan with distinction. He worked in Muhammad Ali Jinnah University as research associate for one year. Currently he is doing MS in Electronic Engineering at Karadeniz Technical University Trabzon, Turkey under Turkish government scholarship program.