Geez Character Recognition Using Deep Convolutional Neural Network

BY
Tesfay Gebrekirstos

ADVISOR
Alhayat Ali(PhD)

CO-ADVISOR
Ephrem Teshale(PhD)

A Thesis submitted to
School of Electrical and Computer Engineering
Addis Ababa Institute of Technology

In Partial Fulfillment of the Requirements for the Degree of Master of Science
(Communication Engineering)

Addis Ababa University
Addis Ababa, Ethiopia
May 20, 2019

Declaration

I, the undersigned, declare that this thesis titled, **Geez Character Recognition using Deep Convolutional Neural Network** and the work presented in it are my own. I confirm that :

- This work was done wholly or mainly while in candidate for a Master of Science degree at Addis Ababa Institute of Technology
- Where I have referred the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given.
- I have acknowledged all main sources of help.

Tesfay Gebrekirstos

Name	Signature

Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering

This is to certify that the thesis prepared by **Tesfay Gebrekirstos**, entitled *Geez Character Recognition Using Deep Convolutional Neural Network* and submitted in partial fulfillment of the requirements for the degree of Master Science (Communication Engineering) complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Internal Examiner	_____	Signature	_____	Date	_____
External Examiner	_____	Signature	_____	Date	_____
Adviser	Alhayat Ali(PhD)	Signature		Date	May 31, 2019
Co-Adviser	Ephrem Teshale(PhD)	Signature	_____	Date	_____

_____
Dean, School of Electrical and Computer Engineering

Dedicated to the loving memory of my father.
1943 – 2011 E.C

ABSTRACT

Optical Character Recognition (OCR) is the process of converting scanned pa-per documents into searchable, computer understandable (digital) documents. OCR shows great potential for rapid data entry with less error prone, scan and preserve historical documents, archives and records management systems, convert scanned documents into searchable text, text to speech conversion and automatic document classification. But, it has had a limited success when applied to the Geez language, even though, there are a bulk of information available in Geez printed form that needs to be converted into electronic form for easy searching and retrieval as per users need. To facilitate this and other office automations development of successful OCR is a valuable input.

A lot of researches has been done in OCR and it is almost a solved problem in some Latin and Arabic languages and OCR systems are in use today. But the use of OCR systems for Geez language is still in its infancy although some developments have been made in recognizing various types documents. Geez character recognition systems face several challenges including large number of characters in the script, visual similarity of most characters and absence of standard prepared datasets. This research investigates current approaches to Geez character recognition problems and innovates relatively new approach to enhance the performance of recognition and classification for this language. We also introduce a dataset for training purposes. The proposed method has less pre-processing steps and out performs the state-of-the-art datasets preparation approaches.

In this paper a Deep Convolutional Neural Network (DCNN) which is a special type of feed forward multilayer Neural network is implemented to recognize all Geez characters and its performance is investigated. Data sets used for training and testing are synthetic multi font Geez characters. Preprocessing and augmentation are applied to increase the number of datasets. The DCNN is trained and tested using the datasets that contain 3751 of Geez characters. Training with different optimization method are applied and the best one with corresponding parameters that increases the performance of CNN is selected. A steady improvement in performance was observed as the number of datasets increased. The proposed method has fewer preprocessing steps and outperforms the standard approach used in classical machine learning techniques. We systematically evaluated the performance of the recognition model and achieved 93% average recognition accuracy (in family level) and 90% average recognition accuracy (in character level). The results are promising and provide a better direction to the future work.

Key Words: Geez Character Recognition, Deep Learning, Deep learning platforms, Convolutional Neural Network, Regularization, OCR, OCRpus

ACKNOWLEDGMENTS

I would like to thank my lovely family and friends which helped me to take big and small decisions in my life and especially during my masters studies.I would like to express my earnest gratitude to Ephrem Teshale (PhD) and Tewodros A.Habtegebrial for giving me the motivation and freedom to work through the challenging task during my master's thesis. I am extremely grateful to Tewodros, for his constant support and belief in me and always encouraging me to pursue my work. I am also thankful to Alhayat Ali(PhD) for his extended support with technical guidance and help.

I would take this opportunity to thank Ephrem Teshale (PhD) for being my academic mentor and supporting me throughout my master's course. I would like to thanks my family and my friends for their immense mental and social support.

There are no adequate words to express my love and gratitude to my beloved wife Hamer Hailezgi who has been by my side throughout the years.Supporting me, encouraging me and carrying more responsibilities during my study. She made me happy during difficult times. My sweet baby Nathan, I would like to tell you that all I do is for you and for a better future for you.

Thank you all, Mom and Bro's.

CONTENTS

1 INTRODUCTION 1
1.1 Background . . . 1
1.2 Motivation . . . 3
1.3 Statement of the Problem . . . 3
1.4 Objective . . . 4
1.5 Methodology . . . 5
1.6 Limitation of the thesis . . . 6
1.7 Outline of the Research : . . . 6

2 RELATED WORK 7
2.1 Classical Recognition Approaches : Previous Local works . . . 7
2.2 Modern Recognition Approaches: International and Local Works . . . 8
2.3 Linguistic Characterstics of Geez - Overview . . . 10
2.3.1 Geez Language . . . 10
2.3.2 Encoding of Geez characters . . . 12

3 TECHNICAL OVERVIEW 14
3.1 Machine Learning . . . 14
3.2 Deep Learning . . . 16
3.3 Artificial Neural Network (ANN) . . . 17
3.3.1 Activation functions . . . 18
3.4 Convolutional Neural Networks . . . 25
3.4.1 Correlation and Convolution . . . 25
3.4.2 Layers . . . 27
3.4.3 ConvNet Architectures . . . 31

4 DATASET AND METHODOLOGY 34
4.1 Dataset Information . . . 35
4.1.1 From Geez Character Unicode . . . 35
4.1.2 By extracting image characters from texts written in Geez characters using OCRpus . . . 37
4.1.3 Data Augmentation on Images . . . 40

5 IMPLEMENTATION AND RESULTS 45
5.1 Geez Character Recognition MODEL . . . 45
5.2 Evaluation Technique . . . 51
5.2.1 Family level Classification . . . 51
5.2.2 Character level classification . . . 52
5.3 Experiment . . . 53
5.4 Experimental Results . . . 54
5.5 Performance comparison . . . 58

6 CONCLUSION, RECOMMENDATION AND FUTURE WORK 62

6.1 Conclusion 62
6.2 Recommendation and future work 62

A APPENDIX 64
A.1 Data loader 65
A.2 Data label reader 65

LIST OF FIGURES

Figure 2.1 Core Geez Characters, Geez numeral, Punctuation and Labialized characters. The 2nd column shows list of basic characters and others are vowels each of which derived from the basic symbol: Source [31] 12
Figure 2.2 Unicode for Ethiopic character set. cells with Grey areas indicate non assigned code points.https://www.win.tue.nl/~aeb/natlang/ethiopic/eth12.png 13
Figure 3.1 Since an early flush of optimism in the 1950's smaller subsets of artificial intelligence first machine learning, then deep learning have created ever larger disruption: [34] 15
Figure 3.2 Illustration of a deep learning model 16
Figure 3.3 Sigmoid 18
Figure 3.4 Tanh 19
Figure 3.5 ReLU 19
Figure 3.6 quantifying loss 21
Figure 3.7 Empirical loss 21
Figure 3.8 Binary cross entropy 21
Figure 3.9 Figure to illustrate back propagation w.r.t connection weight W_2 23
Figure 3.10 Dropout 24
Figure 3.11 Relay on multiple features 24
Figure 3.12 The convolution operation 26
Figure 3.13 The convolution operation 27
Figure 3.14 6 convolution filters output 28
Figure 3.15 Figure showing an example of convolution process.The parameters used is this example are: stride = 2, convolution filter window size = 3, zero pad = 1, depth of layer = 2. Source:[37] 30
Figure 3.16 Example to show how the Max-Pool layer of 2*2 window with 2 strides perform the feature space reduction Source:[37] 31
Figure 3.17 Example to show how the Max-Pool layer of 2*2 window with 2 strides perform the pooling operation. 32
Figure 3.18 A convolutional neural net for labeled images, the architecture is shown in the top panel and the parameters (which are filters) are shown in the bottom panel. 33
Figure 4.1 Unicode for Ethiopic character set. cells with Grey areas indicate non assigned code points.https://www.win.tue.nl/~aeb/natlang/ethiopic/eth12.png 36
Figure 4.2 First 71 geez characters generated from geez amharic unicode for the jirret font type 38

Figure 4.3 First 71 geez characters generated from geez amharic unicode for the goffer font type . . . 38
Figure 4.4 First 71 geez characters generated from geez amharic unicode for the wookianos font type . . . 39
Figure 4.5 First 71 geez characters generated from geez amharic unicode for the fantuwa font type . . . 39
Figure 4.6 Jirret Fonts . . . 40
Figure 4.7 Fantuwa Fonts . . . 40
Figure 4.8 Pipline of Ocrpy . . . 41
Figure 4.9 foreground image . . . 42
Figure 4.10 Background image . . . 42
Figure 4.11 Parameter Alpha = 0 . . . 42
Figure 4.12 Parameter Alpha = 0.15 . . . 42
Figure 4.13 Parameter Alpha = 0.25 . . . 42
Figure 4.14 Parameter alpha = 0.5 . . . 42
Figure 4.15 Parameter alpha = 0.75 . . . 43
Figure 4.16 Parameter alpha = 1.0 . . . 43
Figure 4.17 Sprite Data sets . . . 44
Figure 5.1 Architecture of a traditional convolutional neural network. . 46
Figure 5.2 The proposed DCNN architecture for Geez character image recognition . . . 48
Figure 5.3 A visualization of the splits . . . 51
Figure 5.4 Family level Training loss . . . 55
Figure 5.5 Family level Testing Accuracy . . . 55
Figure 5.6 Family level Training loss . . . 56
Figure 5.7 Family level Testing Accuracy . . . 57
Figure 5.8 character level Training loss . . . 58
Figure 5.9 character level Testing Accuracy . . . 58
Figure 5.10 Histogram of weights for the last step (family level) . . . 59
Figure 5.11 Histogram of gradient weights for the last step (family level) 59
Figure 5.12 Histogram of biases for the last step(family level) . . . 59
Figure 5.13 Histogram of gradient biases for the last step (family level) 59
Figure 5.14 Histogram of weights for the last step (character level) . . . 59
Figure 5.15 Histogram of gradient weights for the last step (character level) . . . 59
Figure 5.16 Histogram of biases for the last step (character level) . . . 61
Figure 5.17 Histogram of gradient biases for the last step (character level) . . . 61

LIST OF TABLES

Table 2.1 Summary of the number of characters in the Geez script (FIDEL) . . . 12

Table 4.1 Statistics of the used datasets 43
Table 5.1 Summary of Training process in family level 60
Table 5.2 Summary of Training process in character level 60
Table 5.3 Performance comparison of related works. 61

LISTINGS

Listing A.1 Python code to generate data 64
Listing A.2 Python code to load and transform data 65
Listing A.3 Python code to load data labels 66

ACRONYMS

ANN Artificial Neural Network

CNN Convolutional Neural Network

CNN Convolutional Neural Networks

CIFAR Canadian Institute For Advanced Research

DIA Document Image Analysis

HMM Hidden Markov Model

MLP Multi Layer Perceptron

MNIST Modified National Institute of Standards and Technology Database

OCRpy Optical Character Recognition Python

OpenCV Open Source Computer Vision

PR Pattern Recognition

ReLU Rectified Linear Unit

SGD Stochastic Gradient Decent

SVM Support Vector Machine

1 INTRODUCTION

1.1 BACKGROUND

It has been a long time after doing research in Optical Character Recognition (OCR) started, specially under the field of pattern recognition (PR). In the last two decades character recognition and text classification is a trending topic in the field of computer vision and machine learning. The capability and process of human reading is the inspiration for developing a machine that could read text with the same proficiency as we humans do. In fact, the very early systems reported for OCR were intended to help the blind to read. Decades of research in this field has resulted in many practical systems, from sorting posts in post offices [1] to automatically recognize bank checks [2], from hand held scanners to sophisticated systems reading the text in natural scenes [3].

Optical Character Recognition (OCR) is a process that allows printed or handwritten text to be recognized optically and converted into machine-readable code that can be accepted by a computer for further processing and can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining [4],[5]. It is a systematic approach used for Geez character recognition by segmenting input image into lines, characters and feature extraction. The extracted features used as an input for a classifier which causes recognition lattice [6] and also need more time and techniques to preprocess and extract feature from the image as well. Over the past decade, Hidden Markov Models (HMM) have received a lot of attention, mainly in the field of speech and handwriting recognition. They have been used for OCR of various printed scripts as well [7], [8]. Though HMMs yield good results, they suffer from a known property of generative models i.e the assumption about the probability of each individual state is independent of its previous states. Moreover, there are many important heuristics that one has to take care of, like HMM structures, input features and character segmentation algorithms.

In order to overcome these shortcomings of pure HMM-based recognition methods, researchers have proposed HMM/ANN hybrid approaches [9]. In these approaches, Artificial Neural Networks (ANN) are used as discriminative feature extractors, while the HMM is used as a text recognition engine. Although, these hybrid approaches perform better than solely HMM based ones, they still suffer from the undesired properties of HMMs [10]. So applying pure artificial neural network models to recognize patterns in image and text is increasing from time to time. For a human being to identify and classify any character is an easy task. For

computers, the classification task is difficult, as computers need a lot of supervision to understand and process an image.

The domain of machine learning algorithms is increasing rapidly. Researchers are applying machine learning algorithms in wide range of problems. There are many different approaches to solve the optical character recognition problem. One of the most common and popular approaches is based on neural networks, which can be applied to different tasks such as pattern recognition, time series prediction, function approximation, clustering, etc. A recent progress in the field of optical character recognition character images category and attribute prediction has been achieved. The task of printed Optical Character Recognition (OCR) is considered a solved issue by many Pattern Recognition (PR) researchers. The notion, however, partially true, does not represent the whole picture. Although, it is true that state-of-the-art OCR systems for many scripts exist. For example, for Latin, Greek, Han (Chinese), and Kana (Japanese), there is still a need for exhaustive research for many other challenging modern scripts [1].

Previously, machine learning algorithms tried to handle the image classification task using images features like the histogram of a gradient (HoG) [11], Scale Invariant Feature Transform (SIFT) [12], color histogram [13], etc. if the images have a very high variation, simple techniques used within traditional machine learning approaches like image feature extraction, histogram of a gradient (HoG)[11], and scale invariant feature transformation (SIFT) [12] do not perform well on image classification tasks. Character recognition from document images that are printed in African scripts is a challenging task, including Geez documents. There are multiple factors that affects development of successful character recognition system some of them are degradation of documents, Printing variations, large number of characters in the script and the presence of noisy data like unnecessary background information, etc. Recently few of the approaches used Deep Convolutional Neural Network (DCNN) models proposed different technique to handle the classification task. Deep CNN works better than traditional machine learning approaches as deep CNN generalizes to address large diversity in the images. Therefore, for image classification tasks most of the researchers moved towards using deep CNN model as it can automatically extract image features and has the ability to learn and adapt.

Furthermore, to make Geez script documents available in the format of electronic text as opposed to page image, we develop an Geez OCR system which is capable to recognize documents written in Geez script with multiple fonts and different level of degradation.

The Geez character recognition task consists of classification subtasks such as family level classification and character level classification. Recent approaches proposed multiple techniques to solve the Geez character classification. We use the ideas from these techniques and propose a different approach to solve the character classification task. In our approach, we use our own synthetic datasets gener-

ated from Geez unicode character using different fonts in a semi automatic way since there are no publicly available dataset for Geez characters. We evaluate the performance of our model for each sub task introduced and also the effect of different algorithms and optimizations methods used. We present a detailed description of the different problems handled and how we address each problem.

1.2 MOTIVATION

Traditionally, transmission and storage of information have been by paper documents though data is entered manually to computers. In the past few decades, documents increasingly originate on the computer. Documents are still printed out for reading, dissemination and markup. The often repeated cry for the need of paperless office has now given way to a different objective, dealing with the flow of electronic and paper documents in an efficient and integrated way. The ultimate solution would be for computers to deal with paper documents as they deal with other forms of computer media. That is paper would be as readable by the computer as magnetic and optical disks are now. If this were the case, then the major difference and the major advantage would be that unlike current computer media, paper documents could be read both by computer and people.

The objective of document image analysis is to recognize the text and graphics components in images, to extract the intended information as a human would. Textual processing deals with the text components of a document image. Some tasks here are: recognizing the text by optical character recognition, determining the skew (any tilt at which the document may have been scanned into the computer), finding columns, paragraphs, text lines and words. Geez OCR research for many of the above mentioned challenges is still in its infancy. Although there many practical OCR systems in use for Latin, Chinese, Arabic languages complex scripts such as Geez scripts lack a reliable OCR software system. The motivation of this study is to use cross knowledge learned from multiple works of these languages to enhance the performance of Geez character recognition since There are no practical OCR systems specifically designed for Geez recognition and classification neither locally nor internationally. Despite the technological advancements in the field of OCR systems, the practical use of such systems is often unsatisfactory. Hence, Preparing standard data sets and developing a model for efficient recognition and classification of Geez characters is necessary.

1.3 STATEMENT OF THE PROBLEM

Developing Geez character recognition systems is a challenging task. Various researchers [14], [15],[16], [17],[18],[19], [20] have been proposed different methods to address the unsolved problem Geez character recognition and a promising result

was found though some of the challenges are still open problems. Some of the key tasks which the proposed system will take into consideration are:

- large number of characters in the script
- Visual similarity in shapes
- unavailability previous developed datasets
- lack of standard representation for the fonts and encoding
- Printing variations

Due to the above problems and limitations development of successful complete OCR software system for Geez characters is difficult and results are poor.Thus absence of locally and/ or internationally developed optical character recognition software for Geez characters in commercial is more or less an open problem until now. To overcome these problems, we propose a deep learning techniques that can be effectively apply to recognizing Geez characters. Deep learning has become ubiquitous for computer vision tasks. Convolutional Neural Networks(CNN) that took over the field and are now the state of the art for object classification and detection. CNN an extremely efficient at extracting meaningful statistical patterns in large-scale and high-dimensional datasets is proposed to address/optimize the problem of Geez character recognition.

1.4 OBJECTIVE

General Objective

The general objective of this thesis work is to design and implement Geez character recognition system using Deep Convolutional Neural Networks (DCNN) which is state-of-the-art for image recognition and classification that turns out less misclassification error for all Geez characters.

Specific Objective

- Study the basic concepts in developing Optical character recognition using deep learning.
- Understand the concept of deep ANN in general and convolutional neural network in particular.
- To understand ConvNet architecture models and implement them.
- To select the best development platform for deep learning algorithms specifically for ConvNets.
- Selecting the proper development environment based on the amount of time to train the network, the libraries that are interfaced to them.

- To train the developed Convnet model and select the trainable network parameters.
- To save and test the model using selected network parameters during training.
- To collect ground truth datasets that for training and testing our model that may help for future Geez character recognition works too.
- Employment of preprocessing algorithms for efficient recovery of characters information especially for old degraded printed documents to increase the variety of datasets.

1.5 METHODOLOGY

The methodology employed consists of collecting/gathering data sets from image of ancient manuscripts that contain Geez characters , generating from Geez unicodes, dataset preparation, implementing data loading and acquisition using data loaders, pre-process and system model design. The important steps that we follow are listed below.

- **Literature Review:** Both local and international articles, research papers, proceedings and materials related to Geez optical character recognition are reviewed to strengthen our works and selecting best and efficient algorithms.
- **Data collection and annotation:** preparing appropriate input data for our model and their corresponding labels which are implemented in a semi automatic way.
- **System Modeling and implementation:** Development of deep convolutional neural network, train the model and test the model for future use.
- **Implementation tool:** A computer with GPU for better learning time installed with Ubuntu operating system using python programming language are the basic requirements. Pytorch which is one of the latest and popular deep learning development platforms is used to develop the ConvNet layers. To visualize the model architecture layers, training progress, biases and connection weight at every training step tensor board is used.
- **Interpretation and Discussion:** The implemented and trained model will be evaluated using classical benchmarking criteria in the literature. For example, error-rate, accuracy, confusion matrix etc. The proposed system will be compared with existing works in the literature and future work is recommended.

1.6 LIMITATION OF THE THESIS

Machine learning is turning data into information. Any Optical character recognition developed using Machine learning algorithm needs data. The quantity and quality of training data directly affects the generalization accuracy of a trainable OCR model. An Ethiopic characters (Geez) includes many characters, numbers and different punctuation marks. Since there are no standard data sets for development of Geez character recognition, preparation of abundant datasets is very challenging task. So the number of datasets collected to train and test our model are very small. Another limitation is data sets for Geez numbers, punctuation marks and hand written characters are not collected and the model is not trained and tested for these datasets. The datasets used to train and test our model are synthetic though our model is real. In this research work complete system of deep Convnets for processing and classification of all Geez characters is done though the number and variety of data sets are limited.

1.7 OUTLINE OF THE RESEARCH :

In the thesis, The following organization is followed:

1. Introduction: Presents the introduction and objectives of this work.
2. Related work: Discusses researches that have been conducted on OCR using old and current approaches and their proposed architecture.
3. Technical Overview: Presents the important concepts and terminologies used Machine learning specifically in the convolutional neural network.
4. Dataset and Methodology: Describes the dataset collection techniques used in this research report. The section also consists of insight of different data preparation methods that can be used for the Geez character classification problems.
5. Implementation and Results: Describes the implementation details and corresponding results along with each experimental implementation.
6. Conclusion, Recommendation and Future work: Describes the key findings of our approach and few possible ideas to extend the work.

2

RELATED WORK

Optical character recognition is a trending topic in the field of computer vision and machine learning. Until recently, there have not been many approaches to solve this problem. The character classification problem is a complex one, because of the deformation in aged documents, due to change in brightness and written materials, similarity in character shapes, large amount of Geez characters. The diversity in the dataset makes the classification task more complex. Because of the close similarity in images, the characters with the same attributes or classes always have different types of images. So, creating a generic representation of any class is very tough in the case of Geez character recognition systems.

There are many different approaches for solving the optical character recognition problem. Researches done in this particular domain use different techniques to handle the character recognition and classification problem. In this section, we will present a detailed explanation of the methodology of few of the recognition and classification approaches that have done before.

2.1 CLASSICAL RECOGNITION APPROACHES : PREVIOUS LOCAL WORKS

There are many different approaches for solving the optical character recognition problem. But one of the most common, efficient, popular and state of the art approaches is based on deep neural networks. In this section, we will review some OCR implementing approaches before the emerging of Deep Convolutional Neural Network(DCNN).

Fitsum demssie [16] states that some students of the departments of information science, computer science and electrical and computer engineering of the Addis Ababa University have produced research output that is of interest to the fields of Optical Character Recognition Software development to Ethiopic documents. And, some international researchers and students from India and Sweden Universities did doctoral researches.

The researches have used different algorithms to reach to a better result in the recognition of characters from scanned and printed documents. Some of the previous international and local works implemented using different algorithms and techniques that we reviewed are:

- **Developing Character Recognition for Ethiopic Scripts** Fitsum Demissie Master Thesis Computer Engineering Programme in Computer Engineering - **Applied Artificial Intelligence Dalarna university** 2011 [16]

- **Recognition and retrieval from document image collections** [21]
- **Recognition of Modification-based Scripts Using Direction Tensors** Lalith Premaratne , Yaregal Assabie and Josef Bigun School of Information Science, Computer and Electrical Engineering Halmstad University, S-301 18 Halmstad, Sweden (IEEE, Recognition of Modification-based Scripts Using Direction Tensors, 2007) [22]
- **Amharic Character recognition System for Printed Real-life Documents using MatLab Tool box** Abay Teshager **Addis Ababa Universty School of Informatics**,2010 [23]
- **Lexicon-based Offline Recognition of Amharic Words in Unconstrained Handwritten Text** Yaregal Assabie and Josef Bigun School of Information Science, Computer and Electrical Engineering Halmstad University, Halmstad, Sweden,2008 [24]
- **Online Handwriting Recognition of Ethiopic Script** Yaregal Assabie and Josef Bigun School of Information Science, Computer and Electrical Engineering Halmstad University, Halmstad, Sweden,2010 [25]

All the above researchers obtain different results and generate different set of datasets but lack of coordinating with other researchers to develop a production quality or commercial OCR beyond the academic exercise is a problem [16].

2.2 MODERN RECOGNITION APPROACHES: INTERNATIONAL AND LOCAL WORKS

Researches that are implemented using deep neural network approaches including Convolutional Neural Networks (for single character level recognition)and Long Short Term Memory (LSTM) techniques for sentence level recognition fall under this category. It is true that state of the art OCR systems for many scripts exist. For example, for Latin, Greek, Han(Chinese) and Kana (Japanese) but there is still a need for exhaustive research for many other challenging scripts[1], [26]. Character recognition from document images that are printed in Geez scripts is a tedious task, due to degradation of documents, printing variation, large number of characters in the script, visual similarity of most characters in the script and multilingual influence in the current globalization[18].

Important research papers for Ethiopic (Geez) and non Ethiopic scripts implemented using state of the art approaches that we have reviewed are listed below:

- **Deep learning for Ethiopian Ge'ez script optical character recognision**Halefom Tekle Weldegebriel and JinXiu Chen Department of Computer Science, Xiamen University Xiamen University, XMU, Xiamen, China . [19]
- **Arabic Handwritten Characters Recognition using Convolutional Neural Network** Ahmed El-Sawy and Mohamed Loey Hazem EL-Bakry Faculty of Computer and Informatics ,Egypt, 2017. [26].

- **Generic Text Recognition using Long Short-Term Memory Networks** Adnan Ul-Hasan Department of Computer Science University of Kaiserslautern, 2016.[1].
- **Ancient Ethiopic Manuscript Recognition Using Deep Learning Artificial Neural Network** By Siranesh Getu Endalamaw Addis Ababa University , Master of Science in Computer Engineering 2016. [27].
- **Artificial Neural Network Approach to the Development of OCR for Real Life Amharic Documents**Abay Teshager Birhanu, R. Sethuraman International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 1, January 2015 , [20].

In the research works revised, character recognition system use different approaches. Supervised or unsupervised training approach, statistical and machine learning or neural network approaches. In the recognition of ancient scripts different mechanism is applied to preserve the morphological nature of the scripts[27]. It is also pointed that special care and treatment should be done not to lose the very important features of the characters because compared to human handwritings they are different in writing style, shape, and material used to write. They also proposed different algorithms for historical documents based on the nature of characters which is very helpful to our work. The second categories show modern recognition system with lot of techniques and efficient algorithms in which advanced deep learning technology under pattern recognition is applied. Comparison of multi layer perceptron and convolutional neural network for recognition and classification of 26 geez charactrers are implemented by [19] and promising results for the future have obtained.

Until now, several international researchers and developers are doing well on their way, but from Ethiopic scripts optical character recognition point of view they did a little and some of the problem that discovered is stated as follows. Previous researchers become to list the basic reasons why there are not tools available to recognize Ethiopic scripts. But the issues are still tedious and unsolved. This shows Ethiopic Geez script recognition is still an active research area.

According to the previous literatures suggestion the basic reasons why Geez character recognition is not solved and developed by local developer or international multilingual OCR tools are [18],[15],[14][20]

- The total number of the characters are large
- The availability of Visually similar characters
- The absence of responsible person who promote the language in the world for the international software developers.
- Luck of cooperation between developers and local researchers.
- The most important and the most significant one is luck of standard fonts.

As Dereje T.[4] proposed binary morphological filtering algorithm for OCR of Amharic typewritten text. He recommended that adopting recognition algorithms which are not very sensitive to the features of the writing styles of characters helps to enhance recognition rate of Amharic OCR.

Million M.[18] conducted research to investigate and extract the attributes of Geez characters written in different fonts and then generalize previously adopted recognition algorithms. Using different test case 49.38%, 26.04% and 15.75% recognition accuracy were registered for WashRa, Agafari and Visual Geez fonts respectively. In addition, Yaregal [28] noted that, font variation is also a problem in designing OCR system for printed Ethiopic documents.

On the other hand, Shatnawi and Abdallah [29] model a real distortion in Arabic script using real handwritten character examples to recognize characters. They used these distortion models to synthesize handwritten examples that are more realistic and achieved 73.4% recognition accuracy. Haifeng Z.[30] proposed a deep learning method based on the convolutional neural networks to recognize scanned characters from real life with the characteristics of illumination variance, cluttered backgrounds, geometry distortion and achieved promising result.

In addition, to best of the researchers knowledge, no attempts has been made for Geez character recognition using deep learning techniques due to scarcity of training dataset. Even though, many researchers have attempted to recognize a small set of Geez character image employing hand crafted features which is usually not robust.

From the experimental result of the revised papers, CNN and Recurrent neural networks (RNN) approaches showed good performance, DNN Particularly Deep Convnets can be extended to Geez and other languages but feed forward back propagation and SVM showed poor performance and requires improvement. Therefore, there is still room for improvement of the printed Geez character recognition using state-of-the-art techniques that can reduce preprocessing steps and computational resource as well.

2.3 LINGUISTIC CHARACTERSTICS OF GEEZ - OVERVIEW

2.3.1 *Geez Language*

The Ethiopic (Geez) script was developed as the writing system of the Geez language, a Semitic language spoken in Ethiopia and Eritrea until the 10^{th} to the 12^{th} centuries. Although the language ceased to be used in vernacular speech (it now serves a liturgical function only), the script is still widely used for writing the Ethiopian and Eritrean Semitic languages such as Tigre, Amharic and Tigrinya. In

some languages, the script is called "fidel", which means "Alphabet", and individual letters are referred to as fidel. The script is believed by many to have derived from the Epigraphic South Arabian script, of Protsinaitic heritage, although there is some dispute surrounding this assertion some also believe it to have descended from Egyptian hieroglyphics. According to the tradition of the Ethiopian Orthodox Tewahedo Church, the script was divinely revealed to Enos, grandson of the first man, Adam. Unlike other Semitic scripts, Geez is written from left to right [31].

Amharic writing system was adopted from that of Geez. Amharic, which belongs to the Semitic language, became a dominant language in Ethiopia back in history. It is the official and working language of Ethiopia and the most commonly learned language next to English throughout the country. Most historic and literary documents are written and documented in this language [17]. The script is unicameral, and has only three combining characters, which are rarely used. Characters don't interact, and the baseline is standard. Geez language have a range of native punctuation. In particular, although words in modern text are increasingly separated by spaces they may be separated by a word space character instead. Geez language also has its own numeric digits, which are used in an additive way, rather than in the way numbers are formed in western text. [31].

There are essentially 33 main syllographs [32], all consonants, in Geez while the rest are essentially those with additional strokes and modifications added on to the main forms to indicate a vowel sound associated with it or to make aural adjustments in the basic consonant sound as shown in figure 2.1. It must be acknowledged also that there are no upper or lower case distinctions in Geez. There are 33 basic characters, each of which has seven forms depending on which vowel is to be pronounced in the syllable without Geez numbers, punctuations marks and Labialized characters. As displayed in table 2.1, Geez has 20 numeral, 8 Punctuation and 30 diacritics.

The Geez writing system has the following basic characteristics [33]:

- Type of writing system is "abugida".
- The symbols are written in a disconnected manner.
- Writing direction is left to right in horizontal lines.
- Each symbol represents a syllable consisting of a consonant plus a vowel. The basic signs are modified in a number of different ways to indicate the various vowels, figure 2.1.
- The lines of text are read left-to-right and the lines are read in a top-to-bottom sequence.
- There is a proportional spacing between characters and the same is true for words.
- There is no standard way of transliterating the Ge'ez script into the Latin alphabet.

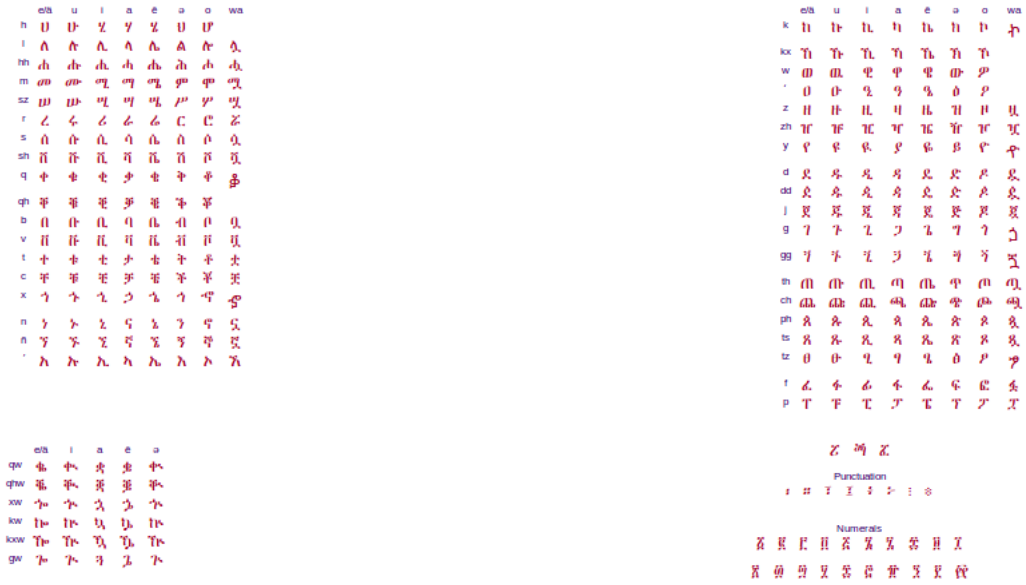


Figure 2.1: Core Geez Characters, Geez numeral, Punctuation and Labialized characters. The 2nd column shows list of basic characters and others are vowels each of which derived from the basic symbol: Source [31]

Summary of Geez character		
No	Type of Amharic Characters	Number of Symbols
1	Core characters	231
2	Labialized characters	51
3	Numerals	20
4	Punctuation marks	8
	Total	310

Table 2.1: Summary of the number of characters in the Geez script (FIDEL)

2.3.2

Encoding of Geez characters

The Ethiopic syllables are originally evolved for writing the Semitic language Geez. Indeed, the English noun Ethiopic simply means the Geeez language. Geez itself is now limited to liturgical usage, but its script has been adopted for modern use in writing several languages of central east Africa, including Amharic, Tigrigna and Tigre.

Encoding Principles: The syllables of the Ethiopic script are traditionally presented as a two-dimensional matrix of consonant vowel combinations. The encoding follows this structure; in particular, the code space range U+1200 up to U+1357 is interpreted as a matrix of 43 consonants crossed with 8 vowels, making 344 conceptual syllables. Most of these consonant-vowel syllables are represented by

characters in the script, but some of them happen to be unused, accounting for the blank cells in the matrix as shown in figure 2.2.

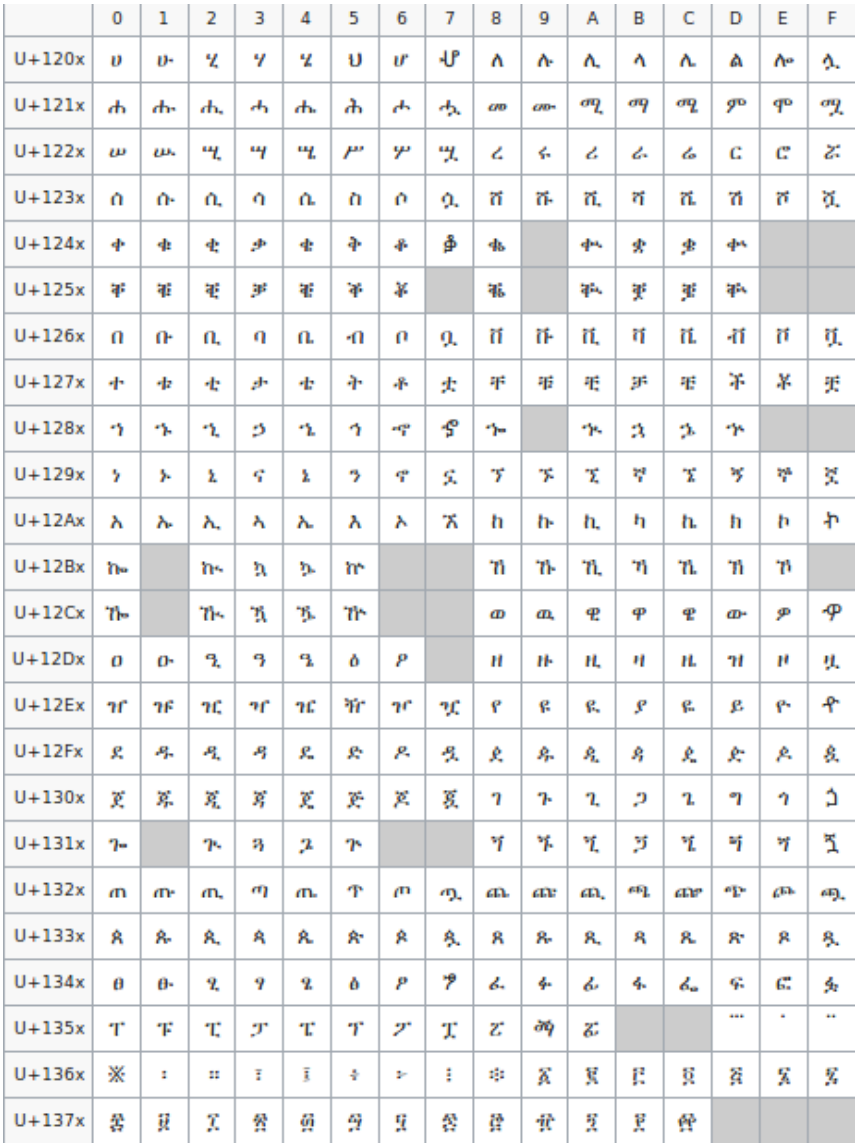


Figure 2.2: Unicode for Ethiopic character set. cells with Grey areas indicate non assigned code points.https://www.win.tue.nl/~aeb/natlang/ethiopic/eth12.png

3

TECHNICAL OVERVIEW

3.1

MACHINE LEARNING

Intelligence is the ability to process information so that they can be used to inform future predictions and decisions. When this intelligence is not engineered but rather by biological inspiration such as in humans it is called human intelligence but when it is engineered it is called artificial intelligence. Artificial intelligence is the study of agents that perceive the world around them, form plans, and make decisions to achieve their goals. Many fields fall under the umbrella of AI, such as computer vision, robotics, machine learning, and natural language processing. Artificial intelligence has amazing success so far and it will shape our future more powerfully than any other innovation this century. Much of our day-to-day technology is powered by artificial intelligence. Building a machine with accuracy close to that of a human expert could greatly increase the quality of life[34].

Machine learning is one of many subfields of artificial intelligence, concerning the ways that computers learn from experience to improve their ability to think, plan, decide and act. Machine learning algorithms can figure out how to perform important tasks by generalizing from examples. This is often feasible and cost effective where manual programming is difficult. As more data becomes available, more ambitious problems can be tackled. As a result, machine learning is widely used in computer science and other fields. Machine learning is already being used in your daily lives even though you may not be aware of it. The amount of data coming at you is not going to decrease, and being able to make sense of all this data will be an essential skill for people working in a data driven industry. Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Machine learning algorithms are split into 3 categories , i.e , supervised learning, unsupervised learning and reinforcement learning. The trends of artificial intelligence are illustrated in figure 3.1.

Machine learning problems where a set of input-target pairs is provided for training are referred to as supervised learning tasks. This is distinct from reinforcement learning, where only scalar reward values are provided for training and unsupervised learning, where no training signal exists at all, and the algorithm attempts to uncover the structure of the data by inspection alone. The nature and degree of supervision provided by the targets varies greatly between supervised learning tasks. For example, training a supervised learner to correctly label every pixel corresponding to an character in an image requires a much more informative target than simply training it recognize whether or not an character is present. To

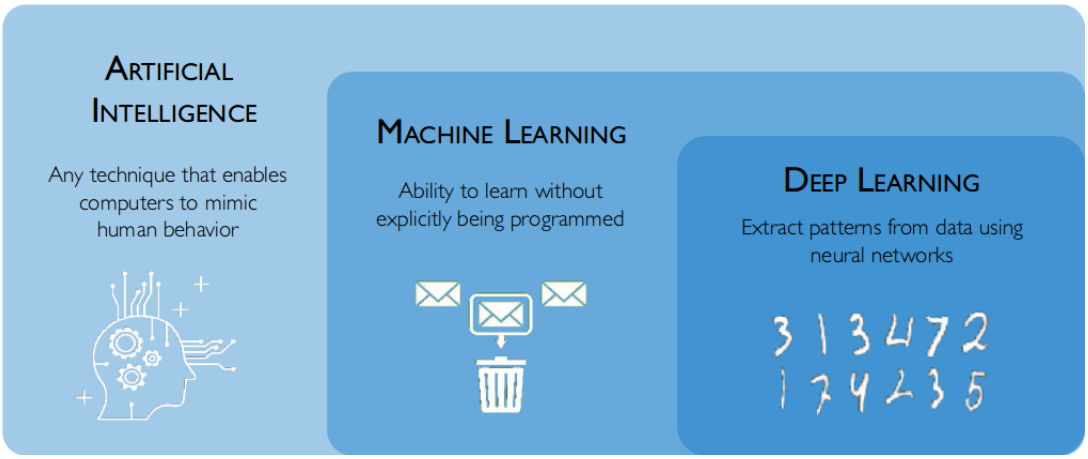


Figure 3.1: Since an early flush of optimism in the 1950's smaller subsets of artificial intelligence first machine learning, then deep learning have created ever larger disruption: [34]

distinguish these extremes, people sometimes refer to weakly and strongly labeled data.

Machine learning grow out of work of AI and deals with the problem of extracting **features** from data so as to solve many different **predictive** tasks.

- Database mining (e.g Web click data, medical records, biology, engineering)
- Forecasting(e.g energy demand prediction,finance)
- Imputing missing data (e.g Netflix recommendations)
- Detecting anomalies(e.g Security, fraud, virus mutations)
- Classifying (e.g credit risk management, cancer diagnosis, image classification, object recognition)
- Ranking (e.g google search, personalization)
- Summarizing (e.g News zeitgeist, social sentiment)
- Decision making (e.g. AI, robotics, computer tuning, clustering, trading)

There are many areas that need application of machine learning:

- Areas where Human expertise is absent(e.g navigation to mars)
- Humans are unable to explain their expertise (e.g speech recognition, vision, language)
- solution changes with time (e.g tracking, temperature control, preferences)
- The problem size is vast for our limited reasoning capabilities (e.g calculating web page ranks, matching ads to face book pages) etc.

Deep Learning is a subset of Machine Learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract one.[35] Deep learning allows the computer to build complex concepts out of simpler concepts. Figure 3.2 shows how a deep learning system can represent the concept of an image of a person by combining simpler concepts, such as corners and contours, which are in turn defined in terms of edges. For text recognition deep learning technique learn categories incrementally through its hidden layer architecture, defining low-level categories like letters first then little higher level categories like words and then higher level categories like sentences.

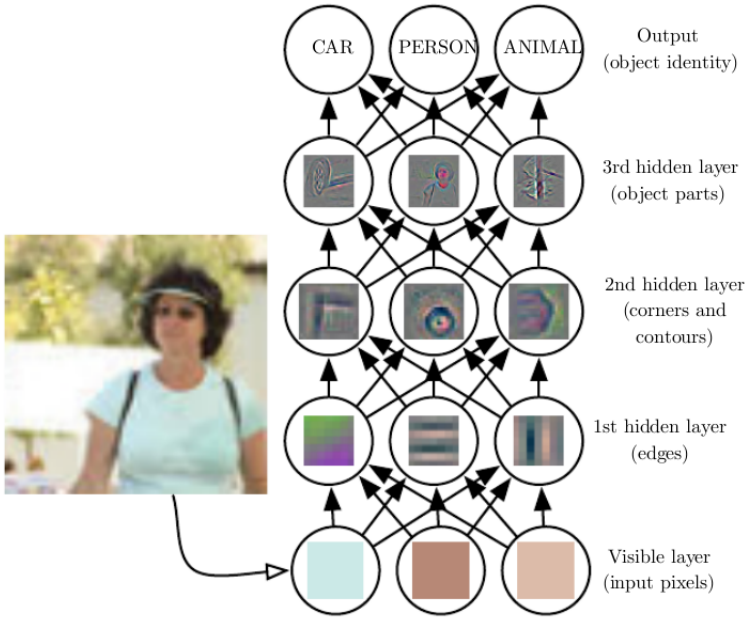


Figure 3.2: Illustration of a deep learning model

Machine learning is the only viable approach to building AI systems that can operate in complicated, real-world environments. Hence, deep learning is really good at learning, particularly in situations where the data is complex. So instead of learning a simple linear model relating input to output, we will instead construct intermediate hidden layers of the network will learn increasingly abstract features, which enables us to not lose all the variations in the complex data. Deep learning is reshaping the world in virtually every domain. In general, deep neural networks are harder to interpret because the features are learned and aren't explained anywhere in any human language. It's all in the machine's imagination unlike shallow neural networks which has small number of layers some times only input and output layers.

A supervised learning task consists of a training set S of input-target pairs (x, z), where x is an element of the input space X and z is an element of the target space Z, along with a disjoint test set S'. We will sometimes refer to the elements of S as training examples. BothS and S' are assumed to have been drawn independently from the same input-target distribution $D_{X\times Z}$. In some cases an extra validation set is drawn from the training set to validate the performance of the learning algorithm during training, in particular validation sets are frequently used to determine when training should stop, in order to prevent over fitting. The goal is to use the training set to minimize some task specific error measure E defined on the test set. Back propagation is one of the error minimization techniques applied on the cost function. The basic concept behind back propagation is to calculate error derivatives. After the forward pass through the net, we calculate the error and then update the weights through gradient descent (using the error derivatives calculated by back propagation).

3.3 ARTIFICIAL NEURAL NETWORK (ANN)

ANN were originally developed as mathematical models of the information processing capabilities of biological brain. The basic structure of an ANN is a network of small processing units or nodes, joined to each other by weighted connections. In terms of the original biological model nodes represent neurons and the connection weights represent the strength of the synapses between the neurons. The network is activated by providing an input to some or all of the nodes, and this activation then spreads throughout the network along the weighted connections. The electrical activity of biological neurons typically follows a series of sharp "spikes", and the activation of an ANN node was originally intended to model the average firing rate of these spikes.

Imagine a linear classification which computes scores for different visual categories given the image using the formula $s = Wx$, where W is a matrix and x was an input column vector containing all pixel data of the image. In the case of digit recognizer where every digit is represented by pixels 32×32, x is 1024×1 column vector, and W is a 10×1024 matrix, so that the output scores is a vector of 10 class scores.

Unlike linear classifier neural network would instead compute $s = W_2 \max(0, W_1 x)$. Here, W_1 could be, for example, a 100×1024 matrix transforming the image into a 100-dimensional intermediate vector. The function $\max(0, -)$ is a non-linearity (Rectified Linear Unit) that is applied element wise. There are several choices we could make for the non-linearity, but Rectified linear unit(ReLU) is the most common choice and simply thresholds all activations that are below zero to zero. All these that adds non linearity are called *activation functions*. Finally, the matrix W_2 would then be of size 10×100, so that we again get 10 numbers out that we interpret as the class scores. Notice that the non-linearity is critical computationally -

if we left it out, the two matrices could be collapsed to a single matrix, and therefore the predicted class scores would again be a linear function of the input. The non-linearity is where we get the wiggle. The parameters W_2, W_1 are learned with stochastic gradient descent, and their gradients are derived with chain rule (and computed with backpropagation).

A three-layer neural network could analogously look like

$$s = W_3 \max(0, W_2 \max(0, W_1 x))$$

Where all of W_3, W_2, W_1 are parameters to be learned. The sizes of the intermediate hidden vectors are hyperparameters of the network. As we increase the size and number of layers in a neural network, the capacity of the network increases. That is, the space of representable functions grows since the neurons can collaborate to express many different functions. Neural networks with more neurons can express more complicated functions. However, this is both a blessing (since we can learn to classify more complicated data) and a curse (since it is easier to over fit the training data). So there is a trade of between the choice of hyperparamters and the size of data sets that we have. Neural Networks that has 3 or more hidden layers are called deep neural networks. All The connection weights (learnable parameters) are learned backpropagation algorithm which will be discussed in the next topic.

3.3.1

Activation functions

Activation functions are needed in order to increase the complexity of the neural networks, without them it would just be a linear sandwich of linear functions (multiplying linear functions = linear function)

3.3.1.1

Sigmoid

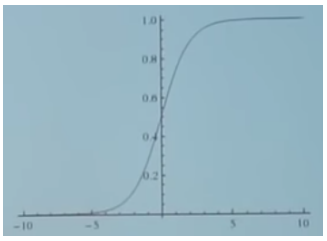


Figure 3.3: Sigmoid

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

- Saturated neurons kill the gradient

- No zero-centered output
- exp() is computationaly expensive

3.3.1.2

Tanh

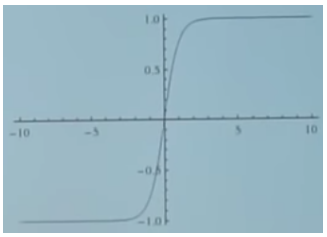


Figure 3.4: Tanh

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

- Zero centered output
- Saturated neurons "kill" the gradient

3.3.1.3

ReLU

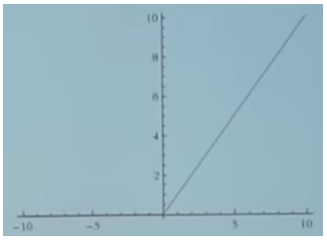


Figure 3.5: ReLU

$f(x) = \max(0, x)$ If a neuron has $x < 0$ the ReLU kills it in back propagation. The gradient will be 0 so that neuron will not back propagation downwards and its weights will not be updated. Its like a boolean gate. The gradient in the case of $x = 0$ is undefined.

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice (6x)
- Not zero centered output
- If your unlucky a neuron may be never active because the initialization has put it outside the manifold.

- When the learning rate is high is easy to kill a lot of neurons. Imagine the activation function as a threshold which is moving while training. If the learning rate is to high it may move out of the data manifold. In that case the neuron dies and will never be able to recover because will never update again.

It is a good practice to initialize them with a slightly positive initial bias to avoid dead neurons. The most commonly used activation function is ReLU, with careful learning rates selection. For example a 3-layer neural network will look like this: $f(x) = W_3 \max(0, W_2 \max(0, W_1 x))$

3.3.1.4

Objective/Cost/Empirical Risk function

The objective function is composed by the data loss expression and regularization term. The data loss takes the form of an average over the data losses for every individual example. That is, $L = \frac{1}{N} \sum_i L_i$ where N is the number of training data. In classification We assume a dataset of examples and a single correct label (out of a fixed set) for each example. The most common choice is the *Softmax classifier* that uses the cross-entropy loss. It transforms the *logits* (scores) output of the network and transforms them into probabilities.

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{3.1}$$

where f_{y_i} is the activation of the correct class and f_j the activation of the class j. In some rough sense the cross-entropy is measuring how inefficient our predictions are for describing the truth. Probabilistic interpretation. Looking at the expression, we see that

$$P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \tag{3.2}$$

Can be interpreted as the (normalized) probability assigned to the correct label y_i given the image x_i and parameterized by W. To see this, remember that the Softmax classifier interprets the scores inside the output vector f as the unnormalized log probabilities (log probabilities means representing probabilities in logarithmic space, instead of the standard [0, 1] interval). This has practical advantages because computers can perform addition more efficiently than multiplication. Exponentiating these quantities therefore gives the (unnormalized) probabilities, and the division performs the normalization so that the probabilities sum to one. In the probabilistic interpretation, we are therefore minimizing the negative log likelihood of the correct class, which can be interpreted as performing *Maximum Likelihood Estimation* (MLE).

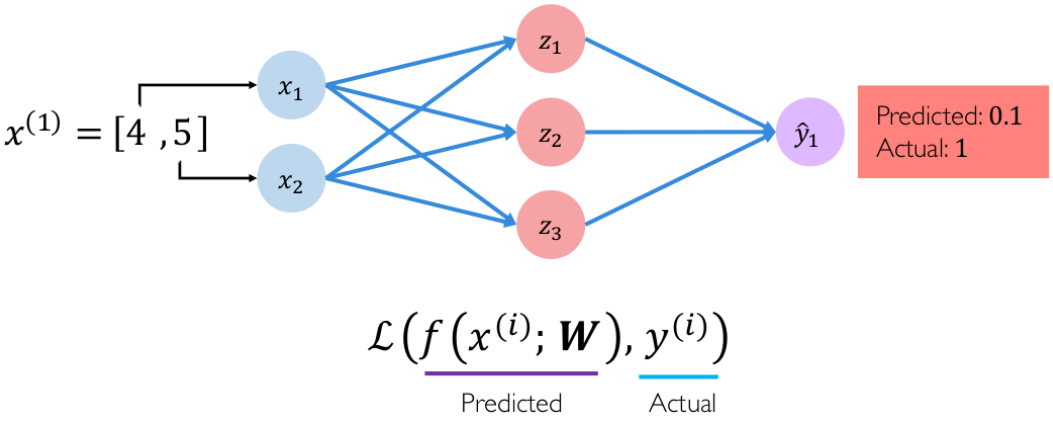


Figure 3.6: quantifying loss

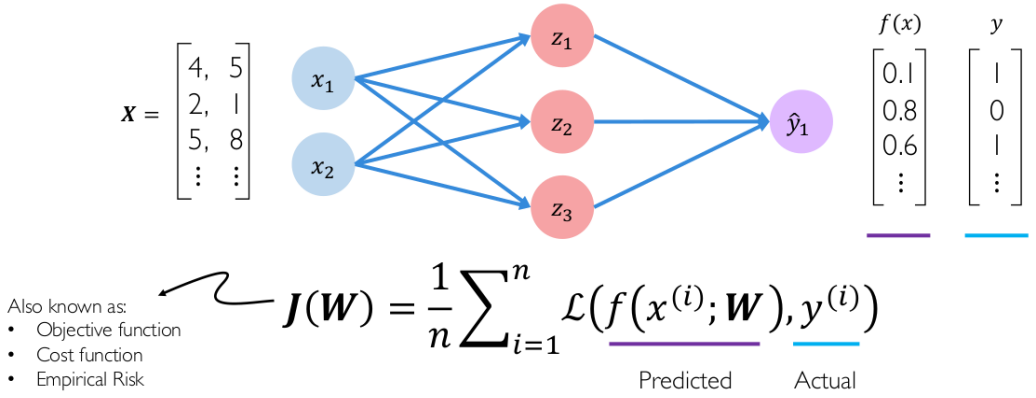


Figure 3.7: Empirical loss

The loss of our network measures the cost incurred from incorrect predictions figure 3.6. The empirical loss measures the total loss over our entire dataset figure 3.7. Binary Cross entropy loss can be used with models that output a probability between 0 and 1 figure 3.8. To optimize the loss we want to find the network

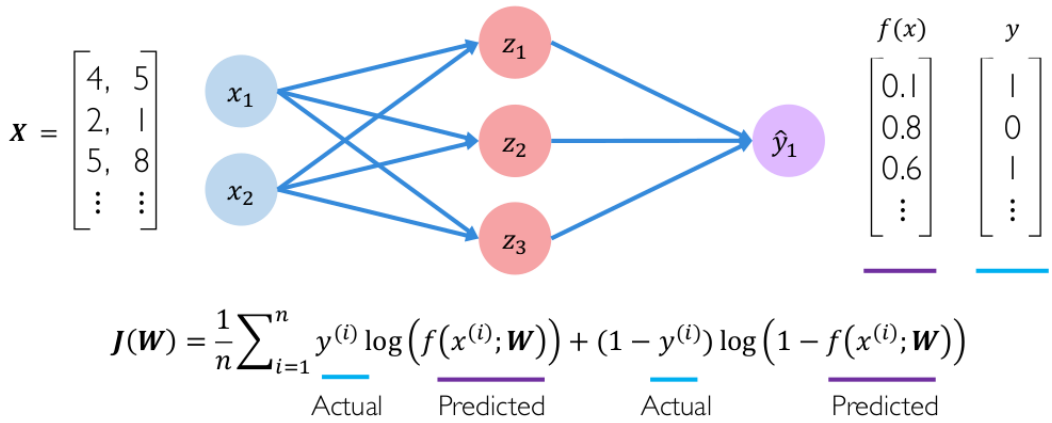


Figure 3.8: Binary cross entropy

weights that achieve the lowest loss.

$$\begin{aligned} \mathbf{W}^* &= \mathrm{argmin}_w \frac{1}{n} \sum_{n=1}^{n} l\left(f\left(x^i; \mathbf{W}\right), y^i\right) \\ &= \mathrm{argmin} J(\mathbf{w}) \\ \mathrm{Remember}, \mathbf{W} &= \left\{W^0, W^1, ...\right\} \end{aligned} \tag{3.3}$$

Remember our loss is function of network weights i.e $\mathbf{W}^* = \mathrm{argmin} J(\mathbf{w})$ so to optimize it randomly pick an initial $\left\{W^0, W^1\right\}$ then compute gradient i.e $\frac{\partial j(w)}{\partial w}$. After calculating the gradient take small step in opposite direction of gradient. Repeat this until convergence.

3.3.1.5

Back propagation

Back propagation helps us to understand how changing the weights and biases affects the cost function. This is achieved by calculating partial derivatives for each weight and for each bias, i.e., $\frac{\partial J(w)}{\partial w}$ and $\frac{\partial J(w)}{\partial b}$. Now, every neuron in the neural network generates some sort of an error. This error affects other neurons and ultimately it affects the global error, meaning it affects our cost function. The middle step of this whole process is calculating this value, and use it to align weights accordingly. By calculating the partial derivate of cost function by the input of each neuron, we define the error:

$$\sigma = \frac{\partial J(w)}{\partial net} \tag{3.4}$$

where net is the weighted input in a certain neuron. In a nutshell, back propagation is happening in two main parts. The First part is called propagation and it is contained from these steps:

- Initialize weights of the neural network
- Propagate inputs forward through the network to generate the output values
- Calculate the error
- Propagation of the output back through the network in order to generate the error of all output and hidden neurons.

The second part of back propagation updates weights of connections:

- The weights output error and input are multiplied to find the gradient of the weight.
- A certain percentage, defined by learning rate of the weights gradient is subtracted from the weight.

for example to know how a small change in one weight for e.g (W_2) affect the final loss $J(W)$ for the figure shown in 3.9 we can compute gradients of back propagation.

$$\frac{\partial J(W)}{\partial W_2} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial W_2}$$

similary by applying chain rule we can get the gradient w.r.t (W_2):

$$\frac{\partial J(W)}{\partial W_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial W_1}$$

Repeat this for every weight in the network using gradients from later layers. Training neural networks by optimizing the loss function some times difficult it needs careful selection proper learning rate. Optimization of loss function using gradient decent is given as follow:

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W} \tag{3.5}$$

where η is the learning rate. Small learning rate converges slowly and gets stuck in false local minimum. Large learning rates overshoot, become unstable and diverge. To get stable learning rates that converge smoothly and avoid local minimum try lots of different learning rates and see what works "just right". But nowadays, there are different adaptive learning rates that adapts to the landscape. Momentum, Adagrad, Adadelta, Adam, RMSProp are examples of adaptive learning rate selection methods.

Figure 3.9: Figure to illustrate back propagation w.r.t connection weight W_2

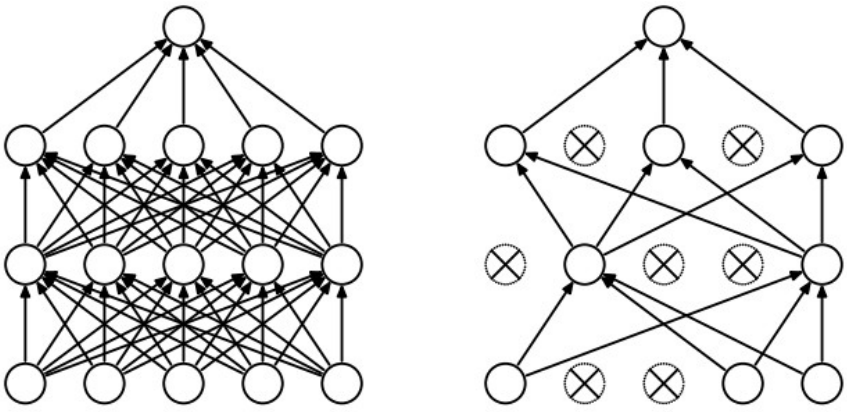
3.3.1.6

Regularizations

Regularization is a technique that constrains our optimization problem to discourage complex models. They improve generalization of our model on unseen data. Drop out and early stopping before we have a chance to over fit are examples of regularizes.

Randomly set 50% some neurons to zero in the forward pass. There two ways of understanding why this works:

- This makes the network unable to relay on a single feature. Thus, forcing it to generate redundant representation. In this way, you will be representing an object with redundant descriptors, detectors etc. So in test time, if some features can not be detected, you still can really on the other ones.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.10: Dropout

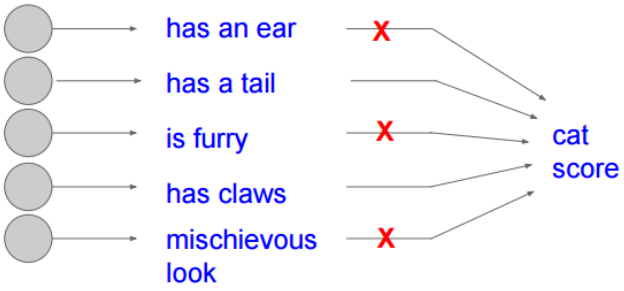


Figure 3.11: Relay on multiple features

- It can also be seen as training a large ensemble of models that share parameters. When applying a mask to shutdown some randomly neurons at the forward pass this mask stays there for the backward pass. Thus, in the back propagation step, no gradient will flow through the neurons that were shut off to zero. So its weights to the previous layer will not be updated. In other words, the neurons that were dropout do not update their connections to the previous layer, just as if they were not there. With dropout you are sub sampling a part of your neural network and training it with the current example. So each binary mask is one model which gets trained on only one data point (or more in the case that two random mask are equal during training).

Deep neural networks provide strategies to solve difficult problems in computer vision. The convolutional neural networks provides an easy method to learn robust features which are invariant to scale, illumination and affine transformations. These features are useful in most of the computer vision applications. In the next section we will discuss the details of CNN which is the core building block of our proposed model for Geez OCR model.

3.4 CONVOLUTIONAL NEURAL NETWORKS

A Convolutional Neural Network (CNN) is a multi layered neural network with a special architecture to detect complex features in data. They have been used in image recognition, powering vision in robots and for self-driving vehicles. CNNs are much more computationally inefficient as compared to traditional neural networks. CNNs were generally used when the problem domain incorporates images. CNNs are designed specifically for taking images as input and are effective for computer vision tasks. Now, it is used for audio and text processing too. CNN became popular after the success of [36] at the ILSVRC challenge. A ConvNet is made up of Layers. Every Layer has a simple API. It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.

3.4.1 *Correlation and Convolution*

In the ConvNet model we start with a convolution layer where the input data are images in two dimensional arrays of pixels. The parameters for the layer are filters which are small two dimensional arrays. The network also employs pooling layers where the image data is sub-sampled so as to reduce the number of pixels in the data. After a series of convolutions followed by pooling the data is passed to a multi Layer perceptron for final classification.

Given a 1D signal x, we can correlate it with a smaller filter, $\mathbf{w}$, of odd length indexed by $-k \leqslant j \leqslant k$ by computing a new signal $\mathbf{z}$. The equation for the convolution operation is shown in equation 3.6. An example to show the correlation operation on sample data is shown in figure 3.15

$$z_i = \sum_{j=-k}^{k} x_{i+j} w_j \tag{3.6}$$

which we write using vector notation as :

$$z = x \bigoplus W \tag{3.7}$$

There are issues at the boundary of the input signal x, but they can be resolved using zero padding. Correlation can be thought of as a matching operation. When the input signal matches the filter then the output signal is high. Convolution is similar to correlation except that we flip the filter about its midpoint before computing the output. So for convolution, if $\bar{w} = \mathtt{flip}(w)$, we compute:

$$z_i = \sum_{j=-k}^{k} x_{i+j} w_{-j} = \sum_{j=-k}^{k} x_{i+j} \bar{w}_j \tag{3.8}$$

which we can write using vector notation as

$$Z = x \bigoplus \bar{w} \tag{3.9}$$

Note that when the filter is symmetric about its mid-point, convolution and correlation are identical.

Convolution can be carried out in more than one dimension. In two dimensions the input and output signals are images and the filters are small two dimensional arrays with a central element. The flip operation is carried out in both dimensions, and the convolution operation becomes:

$$Z_{ij} = \sum_{k1=-k}^{k} \sum_{k2=-k}^{k} x_{i+k1,j+k2} w_{\bar{k1},k2} \tag{3.10}$$

which we write using matrix notation as:

$$\mathbf{Z} = \mathbf{x} \bigoplus \bar{\mathbf{w}} \tag{3.11}$$

We can think of convolution as the sliding of the flipped filter over the input image and at each pixel in the input image taking the dot product of the filter with that piece of the input image currently under the sliding filter. The process is depicted in figure 3.12

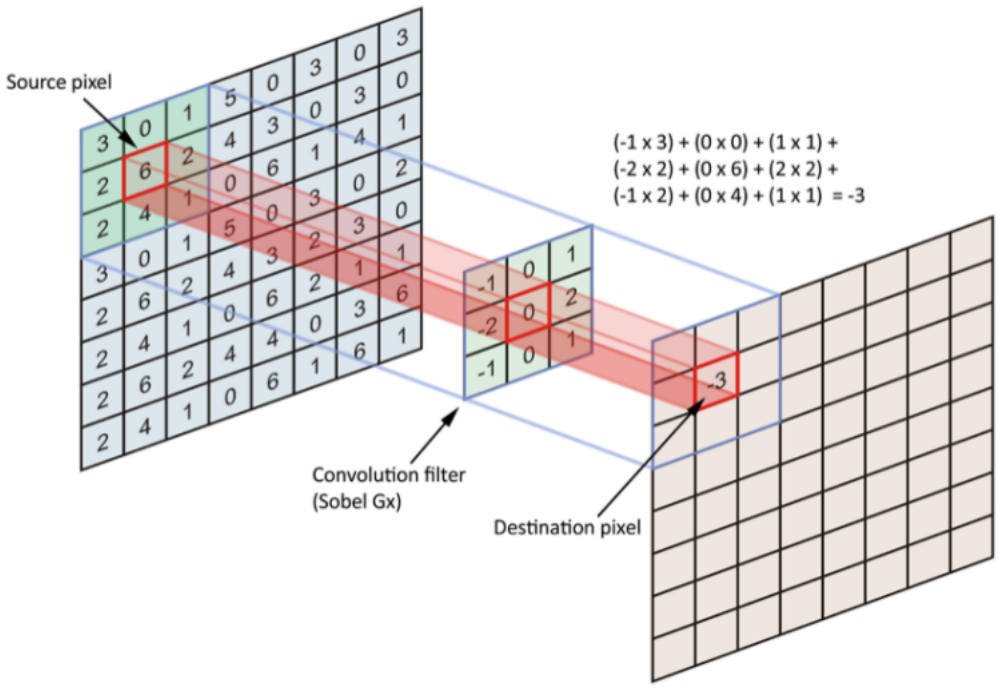


Figure 3.12: The convolution operation

In three dimension input to a convolutional layer of a ConvNet are usually volumes of images, for example an RBG image can be thought of as a 3 dimensional

volume with width **W** , height **H** and depth **3**. The first layer of the network might perform convolutions with say **F** different filters that are also volumes each of size $K \times K \times 3$ The depth of the filters is the same as the depth of the input volume. The filters are translated over the input volume in the width and height dimensions and 3 dimensional dot products are computed in the overlap region to end up with a $W \times H$ output image per filter. So we end up with a $W \times H \times F$ output volume, see figure 3.13 for a pictorial representation.

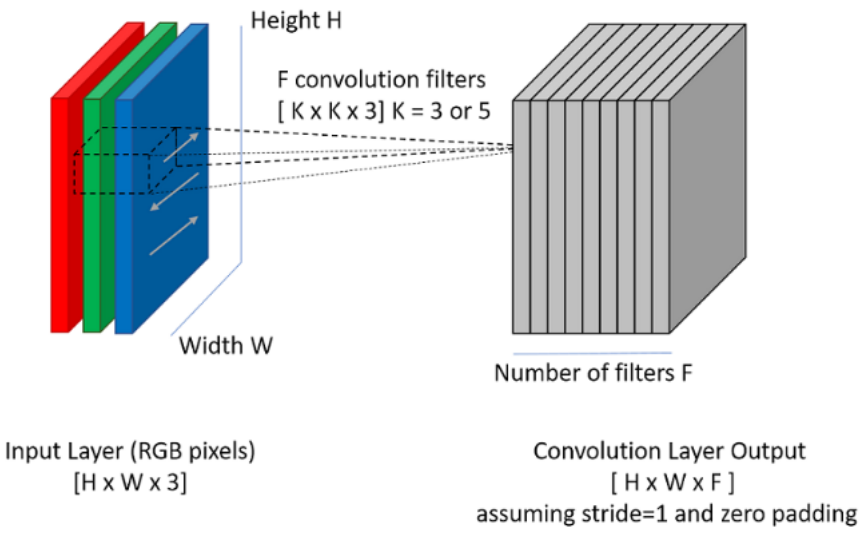


Figure 3.13: The convolution operation

3.4.2

Layers

The architecture of CNN comprises of distinct layers that are arranged at multiple stages. A ConvNet is a sequence of layers and every layer of a ConvNet transforms one volume of activations to another through a differentiable function [37]. We use three main types of layers with out considering the input and output layers to build ConvNet architectures: Input layer, Convolutional Layer, Pooling Layer, and Fully-Connected Layer then we stack these layers to form a full ConvNet architecture. The key components of CNN are listed below along with a brief explanation.

3.4.2.1

Input layer

The input layer (that contains the image) should be divisible by 2 many times. Common numbers include 28 (e.g MNIST), 32 (e.g. CIFAR-10), 64, 96 , or 224 (e.g. common ImageNetConvNets), 384, and 512.

3.4.2.2

Convolution Layer

Convolution layer consists of trainable weights and bias which are shared within a feature layer. These weights are connected to the local regions (receptive fields) in the input image. Every layer has a unique set of weight which computes some output via dot product between weights and the input information. Multiple convolution layers capture different information from the input and these are the main feature extraction units in the CNNs. In a convolution layer we have filters. Each filter is a weight matrix which is convoluted over the input volume. Each time the filter is applied it outputs a single value. As we can show in figure 3.14 Instead of having only one filter, we have a stack of filter which produce a stack of activation maps (one for each filter). In this example this Conv layer has a stack of 6 filters $5 \times 5 \times 3$. All positions in the same activation map share the same weights (filter)

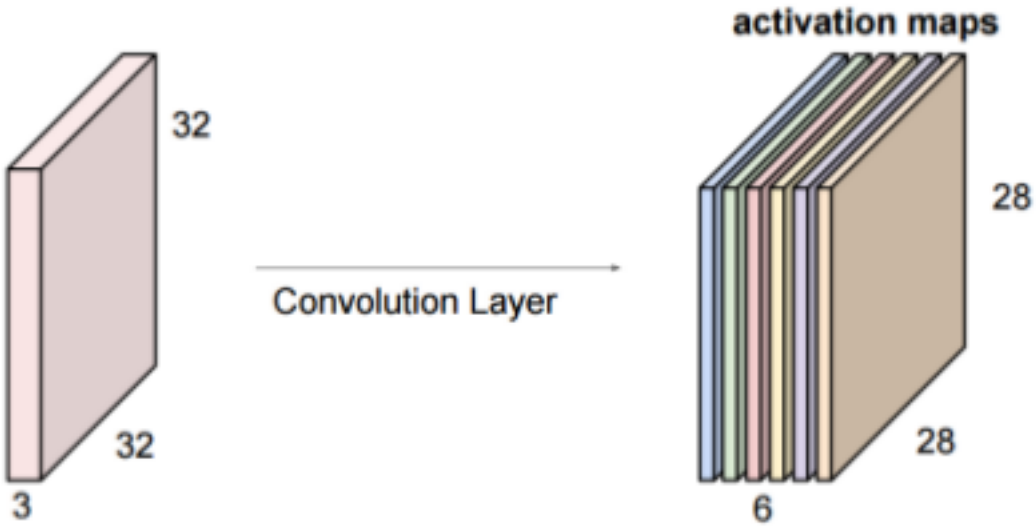


Figure 3.14: 6 convolution filters output

The Convolutional layer accepts a volume of size $W_1 \times H_1 \times D_1$ requires four hyper-parameters, number of filters(F), their spatial extent(K), the stride(S), the amount of zero padding (P), Then it produces $W_2 \times H_2 \times D_2$ where :

$$\begin{aligned} W_2 &= \frac{W_1 - K + 2P}{S + 1} \\ H_2 &= \frac{H_1 - K + 2P}{S + 1} \\ D_2 &= F \end{aligned} \tag{3.12}$$

Depth of any convolutional layer is the equal to the number of feature filters in a layer.

Stride defines the number of pixels any filter of the convolutional layers jumps between consecutive dot product operations. The higher the number of strides the lesser is the size of the output features from the convolutional layer.

Zero-Padding is used to create a black border around the image (as the values in the border would be 0). This feature of convolutional layer helps to keep the same output image size as to the input image. An example of convolution process considering filter window size, stride, padding and depth layer is shown in figure 3.15.

3.4.2.3

Non-linearity

It is common for convolutional layers to pass their output through a non-linear squashing function before the data reaches the next convolutional layer. Common choices is ReLU units. This non-linear units do not have parameters but do affect the back-propagated gradients.

3.4.2.4

Pooling Layer

ConvNets often use pooling layers to reduce the size of the representation, to speed the computation, as well as make some of the features that detects a bit more robust. It is common to periodically insert a Pooling layer in between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network and hence, to control over fitting. This down sampling operations happens on each activation map independently and the volume depth is preserved. Figure 3.16 explains more about the function of pooling layer.

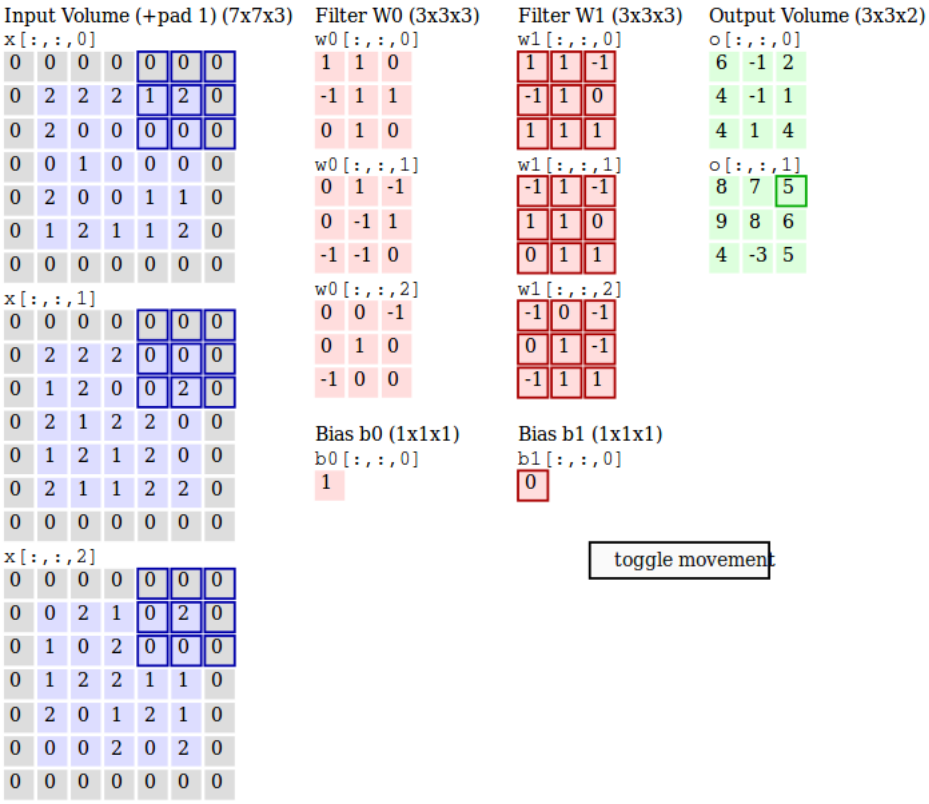


Figure 3.15: Figure showing an example of convolution process.The parameters used is this example are: stride = 2, convolution filter window size = 3, zero pad = 1, depth of layer = 2. Source:[37]

The Pooling layer accepts a volume of size $W_1 \times H_1 \times D_1$ requires two hyper-parameters, their spatial extent(K), the pool stride(S), then it produces $W_2 \times H_2 \times D_2$ where :

$$\begin{aligned} W_2 &= \frac{W_1 - K}{S+1} \\ H_2 &= \frac{H_1 - K}{S+1} \\ D_2 &= D_1 \end{aligned} \tag{3.13}$$

The pooling layer's spatial reduction uses different methods such as the max pooling, average or min pooling of all the values present in the sliding window which iteratively covers all the points on the input feature map. The jumps of the sliding window depend on the stride value of the pooling layer.

The most common pooling layer used is the Max-Pooling layer with a window size of 2 * 2 (finds a max value amount 4 values in feature map) with stride value of 2 (no overlapping pooling window over the feature map).This is depicted in figure 3.17

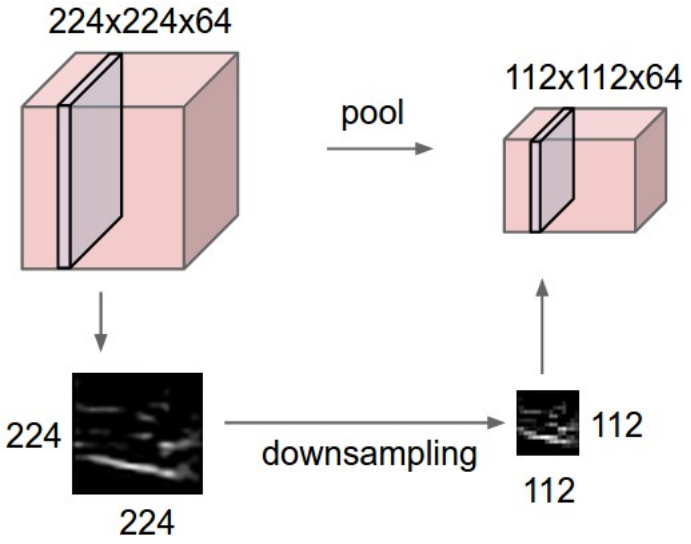


Figure 3.16: Example to show how the Max-Pool layer of 2*2 window with 2 strides perform the feature space reduction Source:[37]

3.4.2.5

Fully Connected Layer(FC)

This layer is similar to layers in the Multi-Layer Perceptron (MLP). Like in the case of MLP, each neuron in the fully connected layer is connected to the each neuron of the next layer. After using multiple convolution and pooling layers, the dimension of the input is reduced, after which we deploy FC layers to compute the class scores. Neurons in a fully connected layer have full connections to all activations in the previous layer, their activations can hence be computed with a matrix multiplication followed by a bias offset.

It is worth noting that the only difference between FC and Conv layers is that the neurons in the Conv layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and ConV layers:

- For any ConvNet there is an FC layer that implements the same forward function. The weight matrix would be a large matrix that is mostly zero except for at certain blocks (due to local connectivity) where the weights in many of the blocks are equal (due to parameter sharing).

3.4.3

ConvNet Architectures

Convolutional Networks are commonly made up of only three layer types: Conv, Pool (we assume Max pool unless stated otherwise) and FC. But we can explicitly

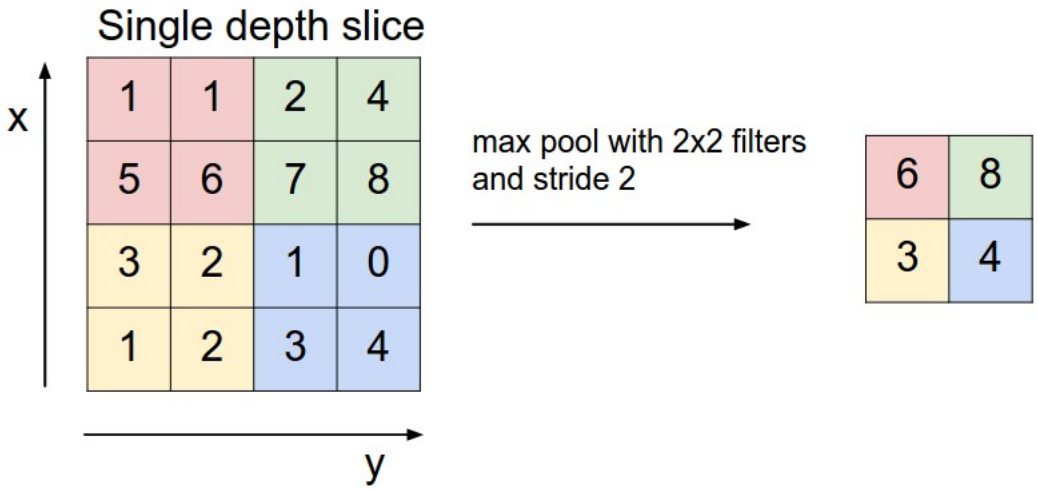


Figure 3.17: Example to show how the Max-Pool layer of 2*2 window with 2 strides perform the pooling operation.

write the ReLU activation function as a layer, which applies element wise non-linearity. This section explains how these are commonly stacked together to form entire ConvNets.

There are five main operations in every ConvNet architectures for what ever application are used:

- Convolution → To extract features from the input image.
- Rectified Linear Unit (ReLU)→ To introduce non-linearity in our ConvNet. It is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.
- Pooling or Sub Sampling →Reduces the dimensionality of each feature map but retains the most important information.
- Classification (Fully Connected Layer)→Every neuron in the previous layer is connected to every neuron on the next layer.
- Softmax function → To map all the final dense layer outputs to a vector whose elements sum up to one.The output layer of a CNN is in charge of producing the probability of each class (each character) given the input image. To obtain these probabilities, we initialize our final Dense layer to contain the same number of neurons as there are classes.

These operations are the basic building blocks of every Convolutional Neural Network. In short the Convolution + Pooling layers act as Feature Extractors from the input image while Fully Connected layer acts as a classifier. A general convNet architectures is shown in figure 3.18.

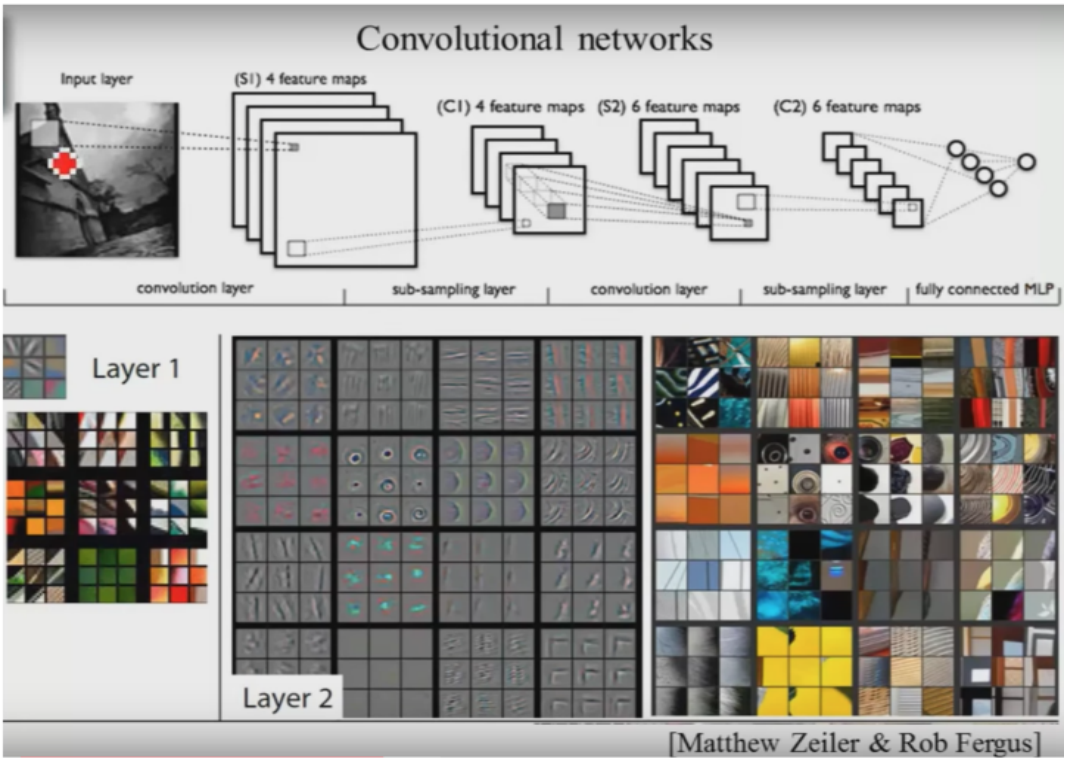


Figure 3.18: A convolutional neural net for labeled images, the architecture is shown in the top panel and the parameters (which are filters) are shown in the bottom panel.

4

DATASET AND METHODOLOGY

Deep learning is taking off and become successful today, this is driven by more powerful computers, larger datasets and techniques to train deeper networks. Hence, In Deep learning preparing large dataset is crucial in achieving highest possible accuracy. Generated data are increased from time to time, this trend is driven by the increasing digitization of society. As more and more of our activities take place on computers, more and more of what we do is recorded. As our computers are increasingly networked together, it becomes easier to centralize these records and curate them into a dataset appropriate for machine learning applications. The age of "Big Data" has made machine learning much easier because the key burden of statistical estimation and generalizing well to new data after observing only a small amount of data has been considerably lightened.

Machine learning all about turning data into information. Hence, the quality and quantity data directly affects the result. Data preparation and processing is an important step that helps make our dataset more suitable for machine learning. And these procedures consume most of the time spent on machine learning. To create a successful Geez character recognition deep neural network model, it is imperative that the developer has the ability to train, test, and validate them prior to deploying the software product for application. Training ,testing and validation must be done with large datasets in deep learning models. The general steps to prepare datasets that are inputs to any machine learning model are:

- *Collect data:* we could collect the samples by scraping a website and extracting data, we could have a device dedicated persons to collect proper data sets, ask to dedicated bodies that prepare a data and deliver to you etc. Most of the time to save some time and effort, one can use publicly available data. But there are no publicly available dataset for Geez character recognition development yet.

- *Preprocess:* Data was changed to a usable format so that we can apply the desired algorithms on all preprocessed data sets. The format that we use in this work is python -tensor. We have used algorithm-specific formatting which are compatible to Pytorch framework.

- *Analyze the input data:* This is looking at the data from the previous task i.e looking at the datasets and to observe some patterns, to extract some features, to plot and visualize them and interpolate the datasets if there are missing data's. But in deep learning this difficulty is solved by breaking the desired complex mapping into a series of nested simple mappings, each described by a different layer of the model to extract abstract features.

- *splitting:* All Geez character image datasets are split in to training, testing and validation sets.

In supervised machine learning, the deep learning model looks at instances of data. Each instance of data is composed of a number of features. Classification, one the popular and essential tasks of machine learning, is used to place an unknown piece of data into a known group(class). In order to build or train a classifier, you feed it data for which you know the class. This data is called your training set. Finally the model (the classifier) is tested and validated by previously unseen data. So in this chapter we are going to discuss how the gathering, preparation and Preprocessing of Geez character image datasets are done.

4.1 DATASET INFORMATION

Data set is very useful to train any classification system and for Geez character recognition system as well. Data set preparation for Geez OCR remain an active area of research till now because there is no available standard data set available to measure the performance of different algorithms.

Data sets used in this research are set of multi-font single Geez character images. The character images were based on 12 different Geez "WashRa" family fonts. Each letter within these 12 fonts was randomly distorted to produce a file containing 3751 character images. Each character images standardized to size 32×32 RGB color of $\mathrm{RGB}(255, 255, 255)$, font size 22 and font color is set to $\mathtt{fontcolor} = (0, 0, 0)$.

In this research data sets are synthetically generated dataset collected from different sources and all data sets are converted and formated to the above mentioned data format. Some sources of datasets used in this work are discussed below :

4.1.1 *From Geez Character Unicode*

Unicode is a universal character encoding standard. It defines the way individual characters are represented in text files, web pages, and other types of documents. Ethiopic character set has got its corresponding unicode based on washra font family. Ethiopic character set unicode is robust and coherent standard for the geez script. They have established a set of governing principles that any standard for the comprehensive Geez character set must meet. These requirements are:

- The members of the set to define the Ethiopic domain must include those characters in both common and infrequent use by publishers of the present day.
- Every member of the domain requires a unique address.
- Characters of a common class be addressed continuously.
- The standard must include a mechanism to facilitate future expansion of the domain.

we have generated Geez character images using 12 fonts (which are member of washra family fonts) from these unicode. The Ethiopic character unicode encoding

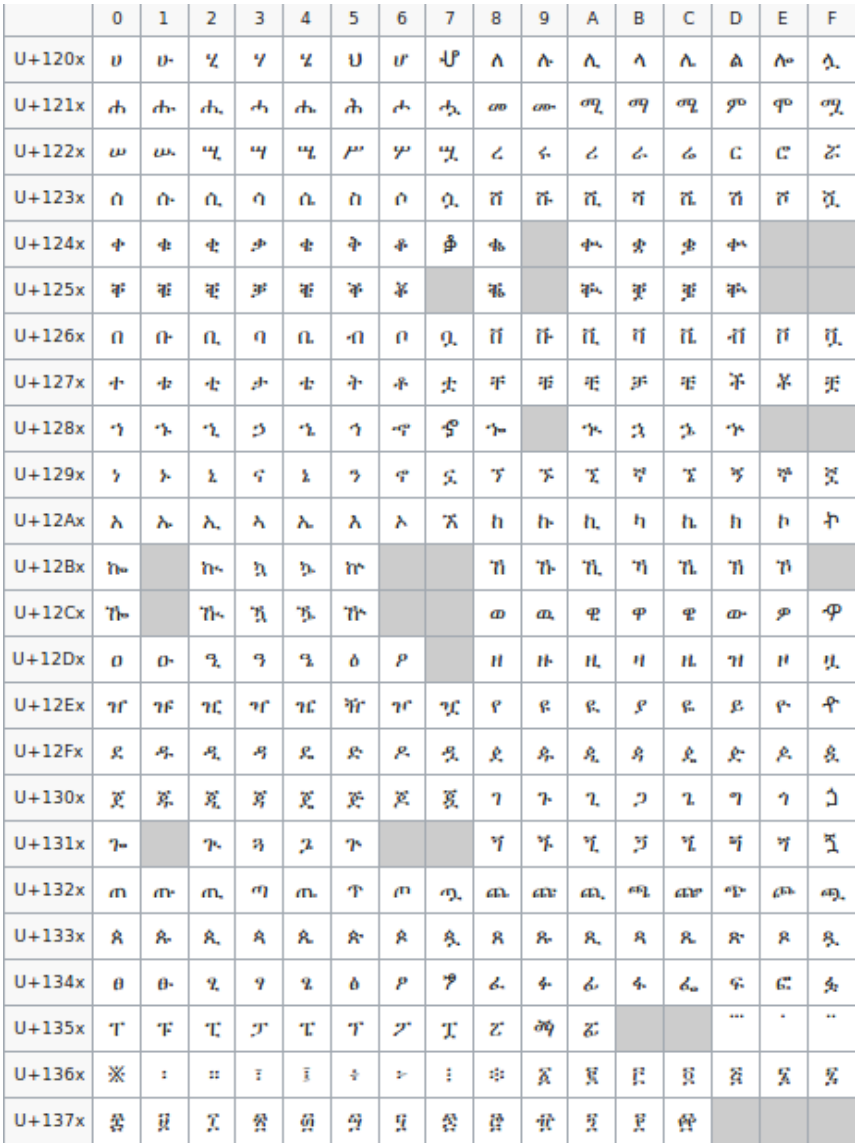


Figure 4.1: Unicode for Ethiopic character set. cells with Grey areas indicate non assigned code points.https://www.win.tue.nl/~aeb/natlang/ethiopic/eth12.png

structure is depicted in figure 4.1.

```
U+1200 TO U+135F =======> Ethiopic Letters
U+1360 TO U+1368 =======> Ethiopic Punctuation
U+1369 TO U+137C =======> Ethiopic Numbers
U+1380 TO U+13BF =======> Ethiopic Extended Letters
U+FDF0 TO U+FDFF =======>  Ethiopic Private Use
```

Data sets are generated from this unicode and characters generated are in a image file format in a semi automatic way. list of basic WashRa font family is given below :

```
Ethiopic WashRa SemiBold: Latin-1, Ethiopic
Ethiopic WashRa Bold: Latin-1, Ethiopic
Ethiopia Jiret: Latin-1, Ethiopic
Ethiopic Zelan: Latin-1, Ethiopic
Ethiopic Hiwua: Latin-1, Ethiopic
Ethiopic Wookianos: Latin-1, Ethiopic
Ethiopic Fantuwua: Latin-1, Ethiopic
Ethiopic Tint: Latin-1, Ethiopic
Ethiopic Yebse: Latin-1, Ethiopic
Ethiopic Yigezu Bisrat Goffer: Latin-1, Ethiopic
Ethiopic Yigezu Bisrat Gothic: Latin-1, Ethiopic
```

Data is generated from all of washra font family listed above. Sample synthetic character image data sets generated from jirret, goffer, wookianos and fantuwa are given in figure [4.2][4.3][4.4] and [4.5] respectively .

4.1.2

By extracting image characters from texts written in Geez characters using OCRpus

We employed an open source generic OCR framework called OCRopus, which is developed by Breuil at DFKI [38]. Ocropus(Ocropy)is a collection of tools for extracting text from scanned images. OCRopus utility requires utf-8 encoded text-lines with the ttf-type font files as an input. Then it generates text line, word, and/or character images with their corresponding ground truth text files [38]. The general process of character image generation along with their ground truth text file is shown in figure 4.8. Furthermore, there are some necessary changes regarding the OCRopus framework source code so as to adopt for documents written in Geez script.

4.1.2.1

Binarization

The first step in the Ocropus pipeline is binarization. The conversion of the source image from gray scale to black and white. Ocropus uses a form of adaptive thresholding, where the cutoff between light and dark can vary throughout the image. This is important when working with scans from books, where there can be variation in light level over the page.

4.1.2.2

Segmentation

The next step is to extract the individual lines of text from the image. There are many ways to do this, Ocropus first estimates the scale of your text. It does this by finding connected components in the binarized image(these should mostly be individual letters) and calculating the median of their dimensions.

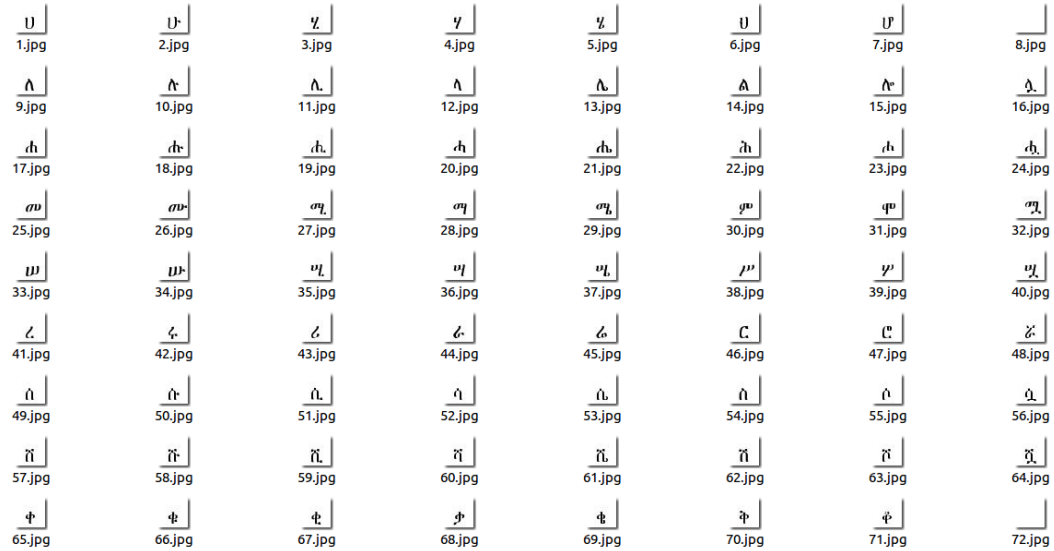


Figure 4.2: First 71 geez characters generated from geez amharic unicode for the jirret font type

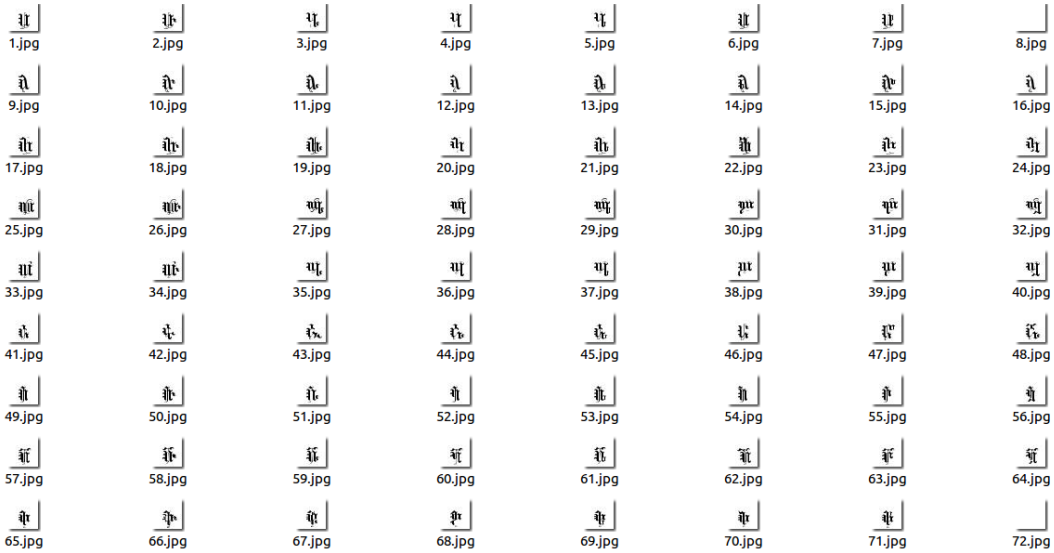


Figure 4.3: First 71 geez characters generated from geez amharic unicode for the goffer font type

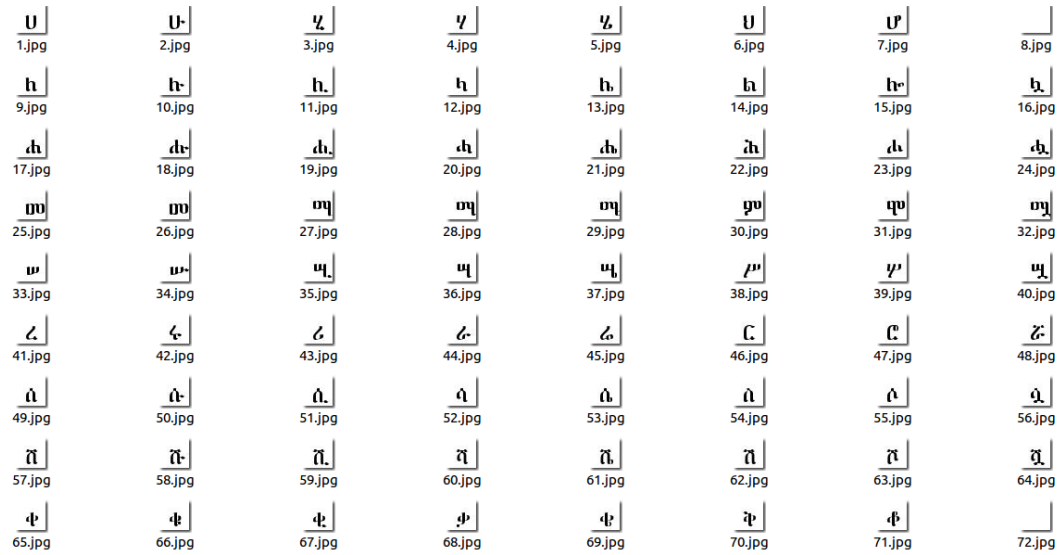


Figure 4.4: First 71 geez characters generated from geez amharic unicode for the wookianos font type

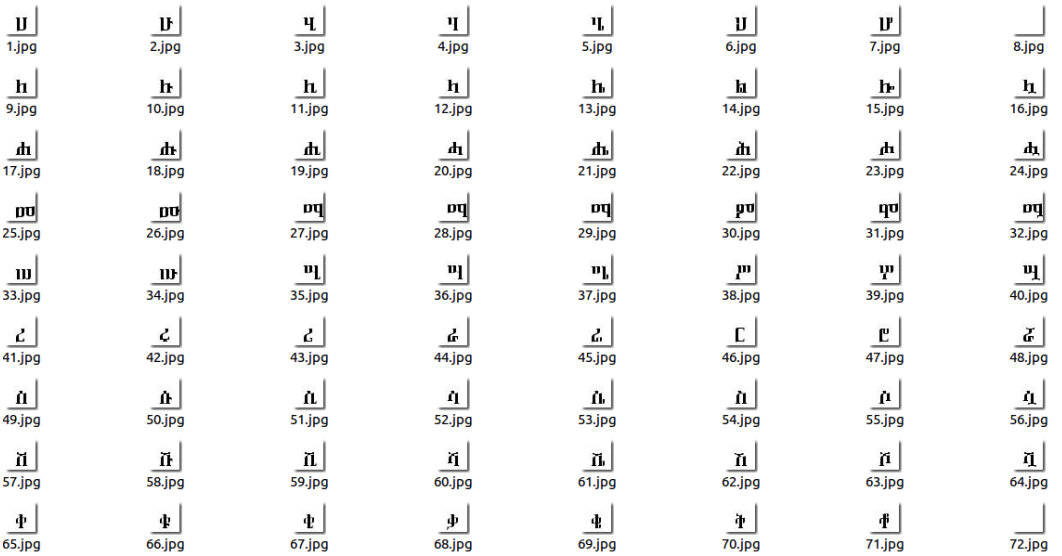


Figure 4.5: First 71 geez characters generated from geez amharic unicode for the fantuwa font type

- It removes components which are too big or too small (according to scale). These are unlikely to be letters.
- It applies the derivative of a Gaussian kernel to detect top and bottom edges of the remaining features. It then blurs this horizontally to blend the tops of letters on the same line together.
- The bits between top and bottom edges are the lines.

By applying OCRpy tool character image data are generated from segmentation results of Geez written text shown in 4.6 and 4.7

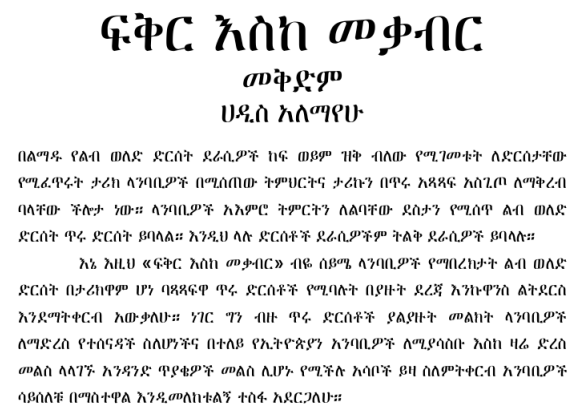


Figure 4.6: Jirret Fonts

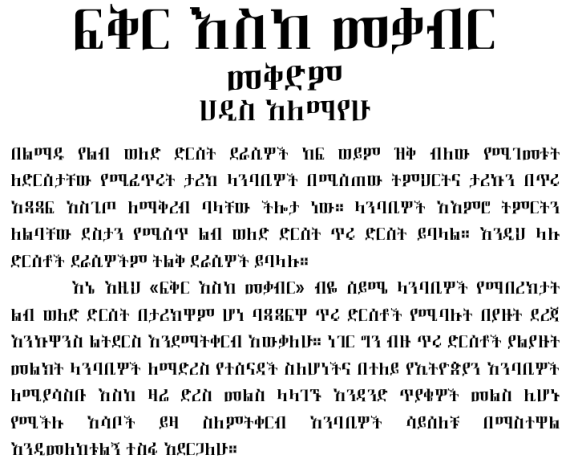


Figure 4.7: Fantuwa Fonts

OCRpy is useful to generate training data for any languages including Geez language. We use Pre-trained models of OCRpy to generate datasets train our model.To generate training data using ocropus-linegen is used. The only input that it takes font type in **ttf file format** and output generates synthetic training data to train our model. Sample characters generated from Ocrpy and the basic pipeline of Ocrpy looks are shown in figure 4.8

4.1.3

Data Augmentation on Images

One of the best ways to improve the performance of a deep Learning model is to add more data to the training set. Aside from gathering more instances from the wild that are representative of the distinction task, we want to develop a set of methods that enhance the data we already have. There are many ways to augment existing datasets and produce more robust models. In the image domain, these are done to utilize the full power of the convolutional neural network, which is able to capture translational invariance. This translational invariance is what makes image recognition such a difficult task in the first place. we want the dataset to be representative of the many different positions, angles, lightings, and miscellaneous distortions that are of interest to the vision task.

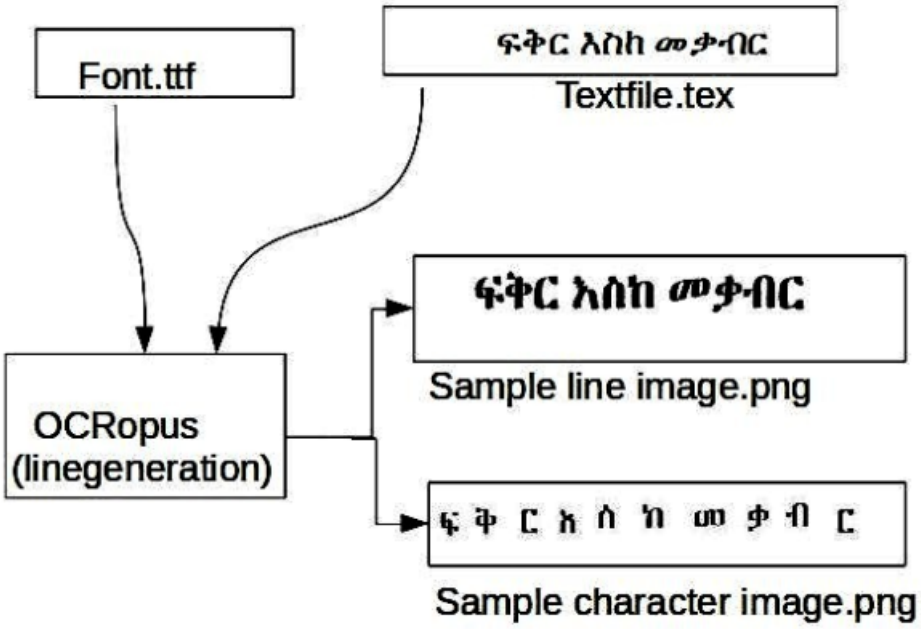


Figure 4.8: Pipline of Ocrpy

There have been researching that present solid proof that with the help of data augmentation, the accuracy of the model can be increased. Another advantage of data augmentation is the prevention of over fitting in the model. To improve the models ability to generalize and correctly label images with some sort of distortion, we apply several translations to our dataset. In our approach, we use following data augmentations.

- **Blending**: Applying arithmetic operation on images.
- **Intensity Change**: In this augmentation, we change the intensity of all pixel value. A random number is selected between [0.8-1.2].For all the pixel values where the intensity value crosses 1, we use to set them to 1.

4.1.3.1

Blending

Arithmetic operations on images like addition, subtraction, bitwise operations etc can be done using OpenCv. Blending (alpha blending) is also image addition, but different weights are given to images so that it gives a feeling of blending or transparency. Images are added as per the equation below:

$$g(x) = (1 - \alpha)\, f_0(x) + \alpha f_1(x) \tag{4.1}$$

By varying α from $0 \rightarrow 1$ we can perform a cool transition between one image to another. Alpha blending is the process of overlaying a foreground image (shown in figure 4.9) with transparency over a background image (shown in figure 4.10).

Figure 4.9: foreground image

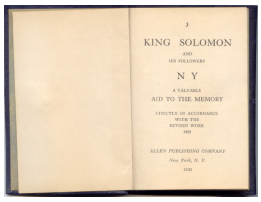


Figure 4.10: Background image

This transparency mask is often called the alpha mask or the alpha matte. Here we took two images to blend them together with different parameter `alpha` as shown in the figure 4.9, 4.10 and get their corresponding results (depending on alpha) as shown in figure 4.11 up to 4.16

Figure 4.11: Parameter Alpha = 0

Figure 4.12: Parameter Alpha = 0.15

Figure 4.13: Parameter Alpha = 0.25

Figure 4.14: Parameter alpha = 0.5

When parameter alpha is **zero** we get the original foreground image character and when parameter alpha is **one** we get the original background image.

Table 4.1 shows the summary of the datasets used for training and testing of our character recognition model.

Deep neural network works when you have large datasets. Some data set of Geez character image datasets that we use for training and testing are shown in the fig [4.17] below. These datasets are ready for the input of our deep CovnNets model in the form of their corresponding binarized image pixel values. In put data should be in tensor data type. Below are some tensor type data sets compatible to our model. **Tensor 0** is the label of the first character's family **(He to Ho)**. And **tensor 1** is label for the character's family **(Le to Lo)**. This label goes until label **tensor 36** for last characetr's family **(Pe to Po)**.

```
tensor(0., dtype=torch.float64)
tensor(0., dtype=torch.float64)
tensor(0., dtype=torch.float64)
tensor(0., dtype=torch.float64)
tensor(0., dtype=torch.float64)
tensor(0., dtype=torch.float64)
tensor(0., dtype=torch.float64)
tensor(1., dtype=torch.float64)
tensor(1., dtype=torch.float64)
tensor(1., dtype=torch.float64)
tensor(1., dtype=torch.float64)
tensor(1., dtype=torch.float64)
```

Figure 4.15: Parameter alpha = 0.75

Figure 4.16: Parameter alpha = 1.0

Datasets	Family classes	Character classes	Height	Width	Depth	Training	Testing
Fantuwua fonts	37	313	32	32	1	275	68
Goffer fonts	37	313	32	32	1	275	68
Hiwua fonts	37	313	32	32	1	275	68
Jirret fonts	37	313	32	32	1	275	68
Jirret slant fonts	37	313	32	32	1	275	68
Tin fonts	37	313	32	32	1	275	68
Washrabold fonts	37	313	32	32	1	275	68
Washraboldfonts	37	313	32	32	1	275	68
wookianos fonts	37	313	32	32	1	275	68
yebse fonts	37	313	32	32	1	275	68
zelan fonts	37	313	32	32	1	275	68
yigezubisratgothic fonts	37	313	32	32	1	275	68
Total number of datasets from all fonts = 3751 Geez character images							

Table 4.1: Statistics of the used datasets

```
tensor(1., dtype=torch.float64)
tensor(1., dtype=torch.float64)
tensor(1., dtype=torch.float64)
```

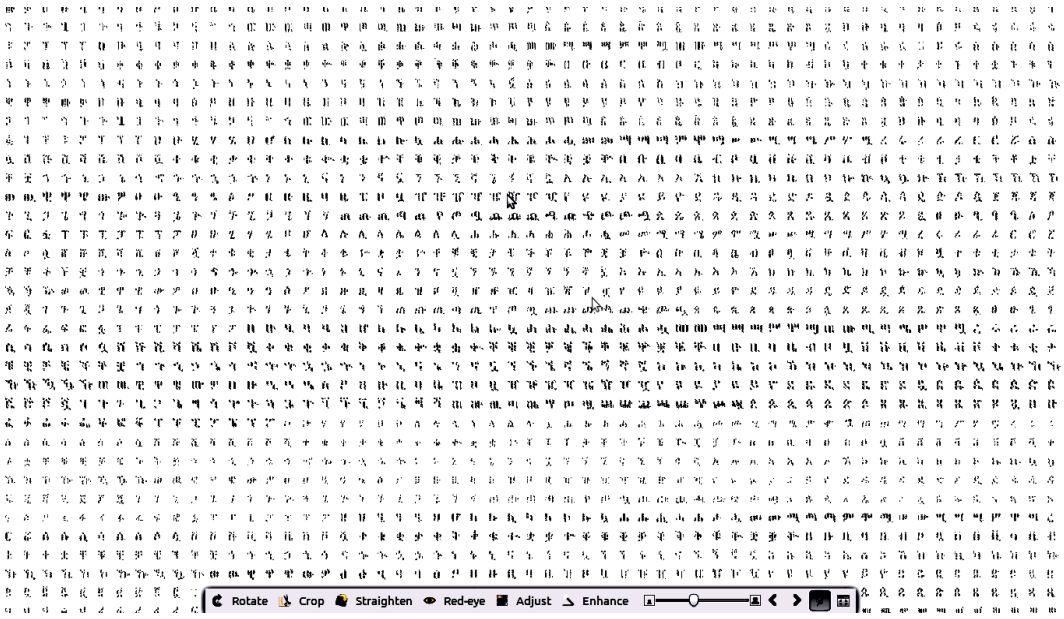


Figure 4.17: Sprite Data sets

5

IMPLEMENTATION AND RESULTS

In this chapter, we discuss the implementation details of our approach. We will discuss the details of all of the success and failure cases. After implementation detail, we will give an overview of our final proposed model and discuss the evaluated results of our model in comparison with the approaches previously discussed in the related works chapter 2.

5.1 GEEZ CHARACTER RECOGNITION MODEL

We start our implementation by taking the category classification task as the preliminary task. Considering the research done in the field of deep learning, MNIST has reproduced one of the best results [39] gives the best results) for the classification task. Pretrained CNN [40] ,[41] model for the MINIST dataset are a proof that CNN is good for classification when input is in image format. We experimentally developed a convolutional neural network model to recognize Geez character images. Convolutional neural network works with raw pixel image as an input rather than predefined features. As we have discussed in chapter 3 convolutional neural network architecture consist of input, Convolutional, Pooling, Relu, fully-Connected layers. There are several variations on the specific architecture; the choices we make are fairly arbitrary. However, the algorithms will be very similar for all variations, and their derivations. The Generic proposed model Deep ConvNets(DCNN) architecture for Geez character image recognition is shown in figure 5.1.

Input: The input layer contains Geez character images with an input size of 32×32 pixel.

Convolutional Layers: The convolution operation essentially performs dot products between the filters and local regions of the input. A common implementation pattern of the ConvNet layer is to take advantage of this fact and formulate the forward pass of a convolutional layer as one big matrix multiply as follows: in our case input is $32 \times 32 \times 1$ and it is to be convolved with $5 \times 5 \times 1$ filters at stride 1, then we take $5 \times 5 \times 1$ blocks of pixels in the input and stretch each block into a column vector of size $5 \times 5 \times 1$. Then we get 832 learnable parameters in the first Conve layer i.e $32 \times 1 \times 5 \times 5 = 800$ connection weights and 32 biases, therefore $800 + 32 = 832$. Then the first convolutional layers output is passed through a non-linear squashing function (ReLU) before the data reaches the next convolutional layer. This layer doesn't have parameters but it affect the back-propagated gradients.

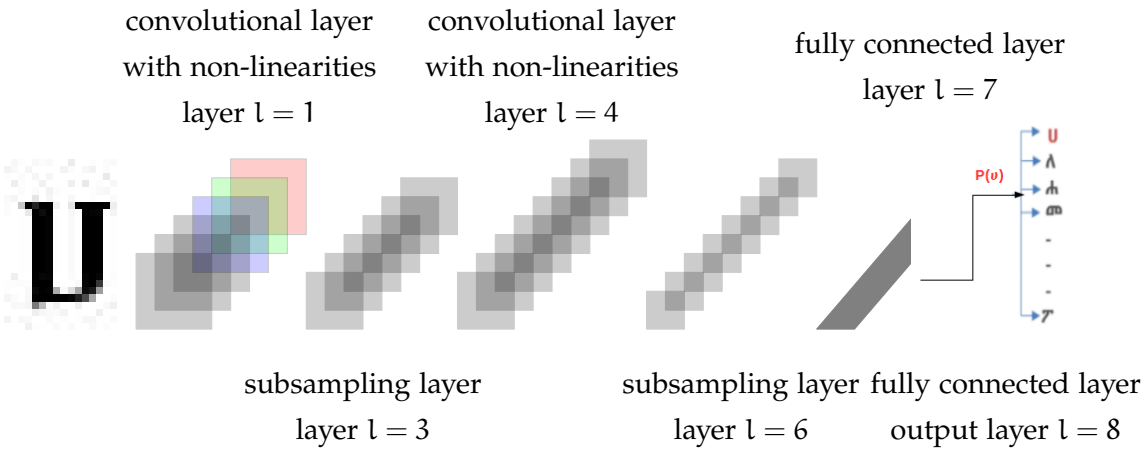


Figure 5.1: The architecture of the original convolutional neural network, as introduced in [42], alternates between convolutional layers, ReLu non-linearities and subsampling layers. In this illustration, the convolutional layers already include non-linearities and thus, a convolutional layer actually represents two layers. The feature maps of the final subsampling layer are then fed into the actual classifier consisting of an 313 of fully connected layers. The output layer usually uses softmax activation functions.

To reduce the spatial size of the representation, to reduce the number of parameters in the network, and hence to also control over fitting max Pooling layer is inserted before two successive Convlayers. To do this a filters of size 2×2 is applied with stride 1 down-samples every depth slice in the input by 2 along both width and height. After first convlayer and non-linearity we get an output shape of $32 \times 1 \times 32 \times 32$ then applying max pooling we get $32 \times 16 \times 16$. It down samples depth slice in the input by 2 along both width and height but the depth dimension remain unchanged.

Finally, after several convolutional and max Pooling layers, the high-level reasoning in the neural network is done via a fully connected layer. A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has. Fully connected layers are not spatially located anymore (you can them as one-dimensional), so there can be no convolutional layers after a fully connected layer. For family level classification the fully connected layer has $37 \times 500 = 18500$. Which can be visualize as one dimensional. Where 37 is the number of classes for the case of family level recognition and 500 is the number of learn able parameters (excluding biases) in the previous layer.

So for our model first convolutional layers dimension can be calculated by applying a convolution C with kernel size 5×5, padding size p = 2x2, stride s = 1x1 on an input map $1 \times 32 \times 1 \times 5 \times 5$ pixel Geez character images, we will get an output feature map. The number of output features in each dimension can be calculated using the formula in 3.12. Similarly the feature learn able parameters of all layers of convolutional layer can be calculated. Some of the choices we need

to make when building a CNN are called **CNN Hyperparameters**. Some of the Hyperparameters are listed below :

- Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, and not using zero-padding would be a narrow convolution. The we have used a padding size of 2 in our model.
- Stride Size: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps. stride size of 1 is used during development of our model.
- Depth : Depth corresponds to the number of filters we use for the convolution operation. One filter of size 5×5 is used in the first Convlayer.
- Channels: Channels are different views of your input data. For example, in image recognition you typically have RGB (red, green, blue) we have 3 channels. You can apply convolutions across channels, either with different or equal weights. The number of channel that applied in the input is 1 .

We created a small python program that calculates the receptive field information for all layers of our CNN architecture. The filter (kernel size), number of padding and strides used in each layer, number of layers, type of activation function used are displayed below in Figure 5.2. It allows us to input the name of any feature map and the index of a feature in that map, and returns the size and location of the corresponding receptive field for each each layers. The following code listing shows an output example of that small python program that shows the over all architecture of our OCR model of Geez character recognition for family level classification since the out put layers has 37 classes or category.

Generally, In each block as shown in Figure 5.1 the outputs of each convolutional layers pass through ReLU activation function followed by single 2×2 pixel window called max-pooling layer. And then three fully connected layers each of with 4,097,000, 500,500 and 18,537 respectively were used at the final layer. The 3×3 convolutional filters with stride 1 was applied for performing filtering and sub-sampling operations 5.2. The total number of parameters used during training of our network can be computed as shown in equation (5.1).

$$P = L^2 \times \left(F^2 \times C^2\right) \tag{5.1}$$

Where P is number of parameters, L is the number of consecutive convolutional layers, F is number of filter and C is channel of the image. We considered an input size of 32×32 pixel Geez character images. The output of each convolutional block is computed using equation (5.2):

$$O = \frac{(W - F + 2 \times P)}{S} + 1 \tag{5.2}$$

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 32, 32]             832
              ReLU-2          [-1, 32, 32, 32]               0
         MaxPool2d-3          [-1, 32, 16, 16]               0
            Conv2d-4          [-1, 64, 16, 16]          18,496
              ReLU-5          [-1, 64, 16, 16]               0
         MaxPool2d-6            [-1, 64, 8, 8]               0
            Conv2d-7           [-1, 128, 8, 8]          73,856
              ReLU-8           [-1, 128, 8, 8]               0
         MaxPool2d-9           [-1, 128, 4, 4]               0
           Conv2d-10           [-1, 256, 4, 4]         295,168
             ReLU-11           [-1, 256, 4, 4]               0
           Linear-12                [-1, 1000]       4,097,000
             ReLU-13                [-1, 1000]               0
           Linear-14                 [-1, 500]         500,500
             ReLU-15                 [-1, 500]               0
           Linear-16                  [-1, 37]          18,537
       LogSoftmax-17                  [-1, 37]               0
================================================================
```

Figure 5.2: The proposed DCNN architecture for Geez character image recognition

Where O is output of each convolutional block, W is input volume, F is kernel size, P is zero padding and S is number of strides. And then the output of each block is fed to the next network block until the probability of the last layer is calculated.

Softmax layer: The purpose of the Fully Connected layer is to use all previous layer features for classifying the input image into various classes based on the training dataset. The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Soft max as the activation function in the output layer of the Fully Connected Layer. The Soft max function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one as shown in equation (5.3).

$$f\left(z_j\right) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{5.3}$$

Where z is the vector of the inputs for output layer and j is the indexes which runs from 1 to k and K is output label.

The general steps that we have followed to develop the entire Geez character recognizer model using Deep ConvNet are listed below:

1 Import Libraries
   - OpenCV: We use openCV to read images files .
   - Tensorboard: To visualize the learning progress and model architecture

2 Prepare Dataset
   - Data sets are prepared as 32*32 character images with 37 labels in family level and 313 in character level.
   - Data sets are splitted in to trainig and testing sets. We have create our own function called split8020(dataset).
   - Size of train data is 80 % and size of test data is 20%.
   - We have develop data set creator and loader module to create feature(pixel images) and labels tensors. And in our model we create variable from these tensors, since variable are used to accumulation of gradients during training.
   - Batch size one of the hyper parameters during traing is the group size used for training. We choose batchsize = 1.
   - Epoch: 1 epoch means training all samples one time. we have used 40 epochs.
   - OCRDataset(): Data set wrapping tensors. Each sample is retrieved by indexing tensors along the first dimension.
   - DataLoader(): It combines dataset and sampler. It also provides multi process iterators over the dataset.
   - Visualization of the images shape in dataset

3 Develop Deep Convolutional Neural network Model in Pytorch frame work.
   - Create feature maps with filters(kernels).
   - Padding: After applying filter, dimensions of original image decreases. However, we want to preserve as much as information about the original image. We can apply padding to increase dimension of feature map after convolutional layer. 2 paddings and 1 strides are used
   - We use 4 convolutional layer.
   - Filter(kernel) size is 5*5

4 Pooling layer:
   - Prepares a condensed feature map from output of convolutional layer(feature map)
   - 3 max pooling layer was applied.
   - Pooling kernel size is 2*2

5 Flattening: Flats the features map. We have used ReLU activation function to do that.

6 Fully Connected Layer:
   - Normal Artificial Neural Network that we know(no shared connection weights) Or it can be only linear like logistic regression but at the end there is always softmax function.
   - No activation function in fully connected layer used.
   - We combined convolutional part and logistic regression to create our CNN model.

7 Instantiate Model Class:
   - create model

8 formulation of Loss function :
   - Cross entropy loss
   - It also has softmax(logistic function) in it.

10 Training the Model
   - For training, we considering Geez character images with an input size of 32×32 pixel are used. To optimize the proposed network model, carried out by using stochastic gradient decent method with batches size of 1 and momentum value of 0.9. A dropout rate of 0.25 was used to regularize the network parameters.
   - SGD Optimizer/any other optimizer are tested and the best optimer depending on the chosen hyper parameters in every training.

11 Prediction or Testing the model

Data Types: The data used with our convolutional network consist of one channel only. Data sets are split to training, validation and testing sets.

- Training Dataset: The sample of data used to fit the model. The actual dataset that we used to train the trainable parameters the model (i.e connection weights and biases). The model sees and learns from this data.
- Validation Dataset: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.
- Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. The test dataset provides the standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets).

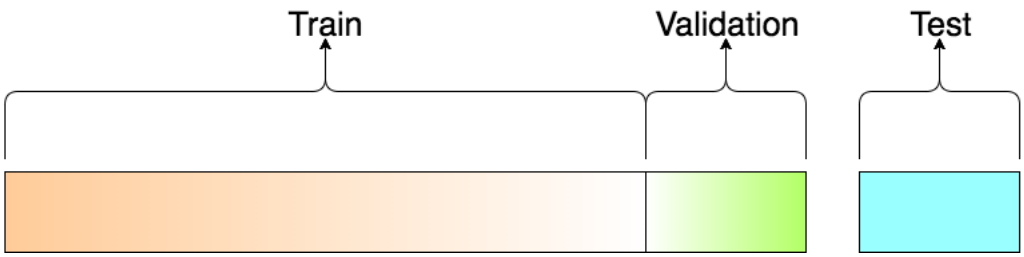


Figure 5.3: A visualization of the splits

The split ratio of the data sets mainly depends on 2 things. First, the total number of samples in your data and second, on the actual model you are training. In our model, we first split the dataset into two: train and test in 4 :1 ratio i.e 80 % of the total data are used as training set. After that, we keep aside the test set, and randomly choose 80 %of the train dataset to be the actual train set and the remaining 20 % to be the validation set, the model is then iteratively trained and validated on these different sets. Which is commonly known as **Cross Validation**.

5.2

EVALUATION TECHNIQUE

We have implemented multi-class and multi-label classification techniques in our model. The evaluation methods for these two task are different. First we have trained and tested the recognition accuracy of family levels and then we train and test it in character level.

5.2.1

Family level Classification

in family level Geez characters are labeled in 37 categories that means we have 37 output of soft max classifier. For family level classification task, the *CrossEntropyCriterion* was used to train the model. Cross entropy cost function

uses the combination of *LogSoftMax* and *ClassNLLCriterion*. This cost function is really well suited for classification with **N**-number of classes and for the imbalanced dataset as well.

Any criterion needs input and target values to train the data. For this criterion, input values should consist of scores for each class and the target values should consist of a single desired or ground truth value. The equation 5.4, explains more about the *CrossEntropyCriterion*

$$\text{loss}(x, c) = -\log\left(\frac{\exp(x[c])}{\sum_{j=1}^{n}\exp(x[j])}\right) = -x[c] + \log\left(\sum_{j=1}^{n}\exp(x[j])\right) \tag{5.4}$$

In This equation n is the number of category present for classification, c is the target value and and x_j is the out put of the model for all category number j.

After training with this criterion, the output from the model for multi-class family level classification is expected to be the likelihood probability of an input for a particular class. For example, if the output of a model for 3 class is 0.25,**0.7** and 0.05, then the model predicts class **2** with, maximum probability, to be the relevant output. As we have mentioned probability, the sum of all the probabilities for N classes is **1**.

Here, the ranked output for category classification is obtained and further evaluated whether the target category values is present on the top or not. Generally, we have 37 classes of same family each contains 8 or 7 character. Therefore, top-8 and top-7 predictions are used to evaluate the multi-class family level classification tasks. If the target value is present in the top-8 predicted values, then we set the top-8 value as true for an input or else false. Similarly, for top-7 prediction,we check the target value in the top-7 output values. Predictions are always sorted in descending order, such that the predicted class with maximum probability is present on the top.

5.2.2

Character level classification

For character level Geez characters are labeled in 313 categories as that means we have 313 output of soft max classifier for this classification task, we use Multi Label Soft Margin Criterion to train the model. Multi Label Soft Margin Criterion's cost function optimizes the combination of Log Soft Max and Class NLL Criterion for multiple labels. The equation 5.5, explains more about the the Multi Label Soft MarginCriterion.

$$\text{loss}(x, y) = -\sum_{j=1}^{n}\left(\frac{y_i \log\frac{\exp(x_i)}{1+\exp(x_i)} + (1-y_i)\log\frac{\exp(x_i)}{1+\exp(x_i)}}{N}\right) \tag{5.5}$$

In This equation N the number of target labels present for an input i, y_i is the target value (either 1 label is present or 0 not present) and x_i is the output of the model.

The outputs are trained in such a way that all the predicted values are ranked such that relevant labels are predicted on top. This strategy helps to rank all the attributes which help in better classification. With the help of ranked results, we can clearly see the predictions made for the input. This ranked result approach was presented in [36], [10] where it was explained how the ranked results help in better classification of a model.

5.3 EXPERIMENT

Once we have designed our model in order to quantify and visualize the performance we conduct an experiment to train , test and validate our model. All experiments were performed on a laptop computer with 8 GB of CPU memory and LINUX operating system. During experimentation we have used 3751 synthetically generated Geez character. We prepared the synthetic datasets by generating from Geez unicode of all Washra fonts. Once we prepare Geez character images, we manually labeled a ground truth set of each character image in a text file. Our model takes these character images and their corresponding labels as an input and then multiple convolutional layers have been adopted to extract automatic discriminating features.

After doing several experiments, we have got a better result with CNN architecture depicted in 5.2. In this architecture, 4 convolutional layers, in each block, we used consecutively with a rectified linear unit (ReLU) as an activation function followed by single 2×2 max-pooling. We also use the Same padding scheme so as to determine the output size and three fully connected layers followed by the Soft max function for the final layer. In this paper, 5×3 convolutional filters with stride of 1 is applied for performing filtering and sub-sampling operations respectively. We did experiments with different architectures of convolutional neural network and the results reported, in this paper, are based on the convolutional neural network architecture illustrated in 5.2.

Once you we have decided and implemented the architecture of the network the second biggest adjusting is the weights (w_i) and biases (b_i) or the parameters of the network. The objective of the training is to get the best possible values of all these parameters which solves the problem reliably. We found the best set of parameters using an algorithm called backward propagation, i.e. we start with a random set of parameters and keep changing these weights such that for every training image we get the correct output. There are many optimizer methods to change the weights that are mathematically quick in finding the correct weights. Stochastic Gradient Descent(SGD) is one such method that we have used in this work. Backward propagation and optimizer methods to change the gradient is a very complicated topic. But we did not need to worry about it as Pytorch and other deep learning frameworks takes care of it.

So, in training process we start with some initial values of parameters and feed 1 training image(in reality multiple Geez character images are fed together) and we calculate the output of the network in a probabilistic manner. Now, we do backward propagation to slowly change the parameters such that the probability of this character image being a a single correct character increases in the next iteration. There is a variable that is used to govern how fast do we change the parameters of the network during training, It is called learning rate (lr=0.001) in our case. If we want to maximize the total correct classifications by the network i.e. we care for the whole training set; we want to make these changes such that the number of correct classifications by the network increases. So we define a single number called **cost function** which indicates if the training is going in the right direction. Typically cost is defined in such a way that as the cost is reduced, the accuracy of the network increases. So, we keep an eye on the cost and we keep doing many iterations of forward and backward propagations till cost stops decreasing.

5.4 EXPERIMENTAL RESULTS

The implementation is based on deep learning framework using python programing on Pytorch Application Program Interface (API) along with torch back end. Pytorch is a Python based scientific computing package (frame work) which is easiest for implementing and constructing deep learning blocks. To visualize the results and network architectures we have used Tensor board. Once the recognition model is developed, experiments were conducted using the data sets. In this paper, our recognition model was developed to classify Geez characetr in 37 family levels the results observed are shown in 5.7 for accuracy of classification and figure 5.6 for loss of classification. After family level classification we also implemented and experimented for character level classification and the observed result is shown in figure 5.9 for character level classification accuracy and 5.8 for character level classification loss.

After training is completed, we visualize results from plots, while loss decreasing, accuracy is increasing and our model is learning (training). Figures 5.6, 5.7 shows loss of the training and accuracy of testing for the family level classification. A representation of the terminal progressive bar for the family level training phase is as below:

```
3751
Iteration: 500  Loss: 3.7004926204681396  Accuracy: 0 %
Iteration: 1000  Loss: 1.597695231437683  Accuracy: 8 %
Iteration: 1500  Loss: 3.0599257946014404  Accuracy: 43 %
Iteration: 2000  Loss: 0.0773080587387085  Accuracy: 40 %
Iteration: 2500  Loss: 1.7244665622711182  Accuracy: 62 %
Iteration: 3000  Loss: 0.6313731074333191  Accuracy: 87 %
Iteration: 3500  Loss: 1.108681082725525  Accuracy: 91 %
Iteration: 4000  Loss: 0.09530112147331238  Accuracy: 93 %
Iteration: 4500  Loss: 0.05109209194779396  Accuracy: 95 %
```

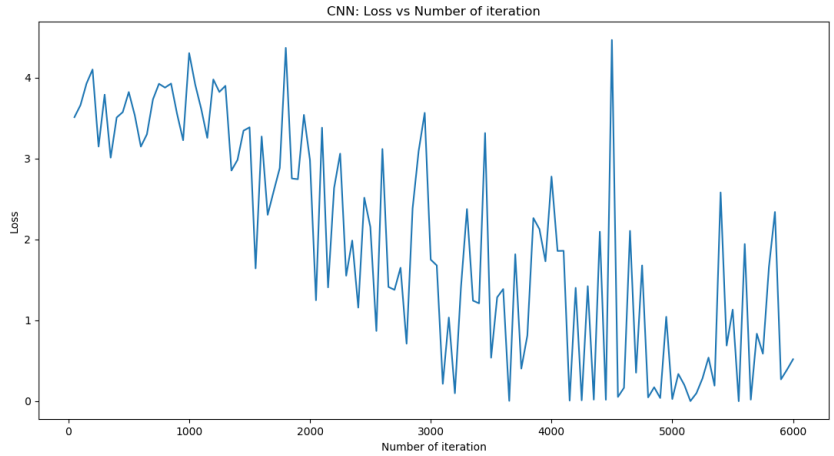


Figure 5.4: Family level Training loss

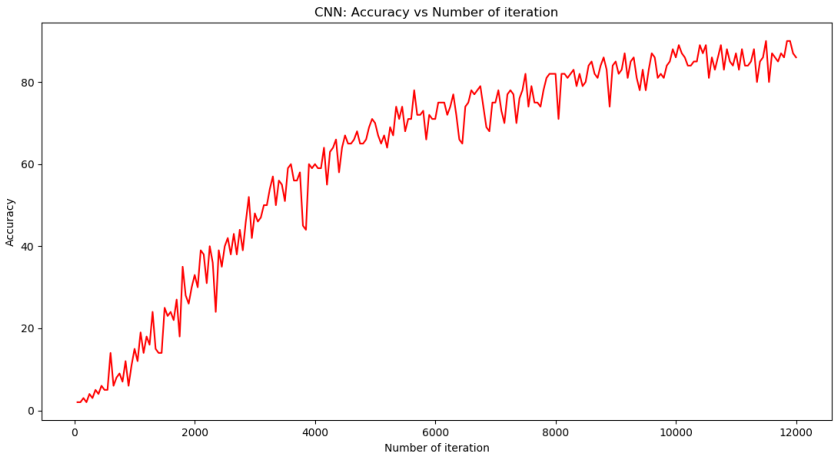


Figure 5.5: Family level Testing Accuracy

```
Iteration: 5000  Loss: 0.001446835813112557  Accuracy: 93 %
Iteration: 5500  Loss: 0.011312856338918209  Accuracy: 93 %
Iteration: 6000  Loss: 0.0010898514883592725  Accuracy: 93 %
Iteration: 6500  Loss: 0.00023500544193666428  Accuracy: 93 %
Iteration: 7000  Loss: 0.017861494794487953  Accuracy: 93 %
Iteration: 7500  Loss: 0.002349600661545992  Accuracy: 93 %
Iteration: 8000  Loss: 0.0017980964621528983  Accuracy: 93 %
Iteration: 8500  Loss: 0.010310914367437363  Accuracy: 93 %
Iteration: 9000  Loss: 0.0004254833038430661  Accuracy: 93 %
Iteration: 9500  Loss: 0.002055299933999777  Accuracy: 93 %
Iteration: 10000  Loss: 0.012136055156588554  Accuracy: 93 %
```

for the above training process(family level) i.e at 1000 iteration we get the following result for loss and accuracy. Similarly, we can visualize results from plots, while loss is decreasing, accuracy is increasing and our model is learning(training).

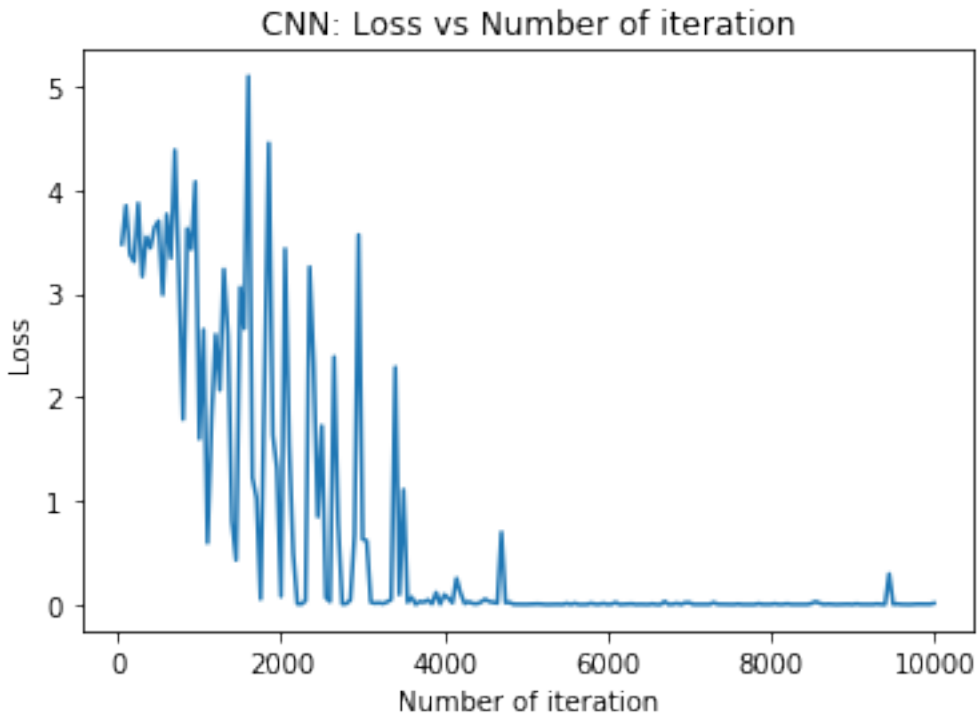


Figure 5.6: Family level Training loss

Figures 5.8, 5.9 shows loss of the training and accuracy of testing for the character level classification. A representation of the terminal progressive bar for the character level training phase is as below:

```
Iteration: 500  Loss: 3.025829792022705  Accuracy: 4 %
Iteration: 1000  Loss: 3.0416088104248047  Accuracy: 30 %
Iteration: 1500  Loss: 0.04989071190357208  Accuracy: 54 %
Iteration: 2000  Loss: 0.4065481126308441  Accuracy: 66 %
Iteration: 2500  Loss: 0.013221027329564095  Accuracy: 66 %
Iteration: 3000  Loss: 0.07309813797473907  Accuracy: 77 %
Iteration: 3500  Loss: 0.002918415702879429  Accuracy: 79 %
Iteration: 4000  Loss: 0.000945672916714102  Accuracy: 87 %
Iteration: 4500  Loss: 0.00031345465686172247  Accuracy: 91 %
Iteration: 5000  Loss: 0.04391350597143173  Accuracy: 91 %
Iteration: 5500  Loss: 0.002313050674274564  Accuracy: 88 %
Iteration: 6000  Loss: 0.002709972904995084  Accuracy: 90 %
Iteration: 6500  Loss: 0.0001737029233481735  Accuracy: 91 %
Iteration: 7000  Loss: 0.0003093659470323473  Accuracy: 90 %
Iteration: 7500  Loss: 0.0016097567277029157  Accuracy: 90 %
Iteration: 8000  Loss: 0.5683431029319763  Accuracy: 91 %
Iteration: 8500  Loss: 0.003260538447648287  Accuracy: 91 %
Iteration: 9000  Loss: 0.0013699972769245505  Accuracy: 90 %
Iteration: 9500  Loss: 0.0004426774103194475  Accuracy: 90 %
Iteration: 10000  Loss: 0.0021088765934109688  Accuracy: 91 %
```

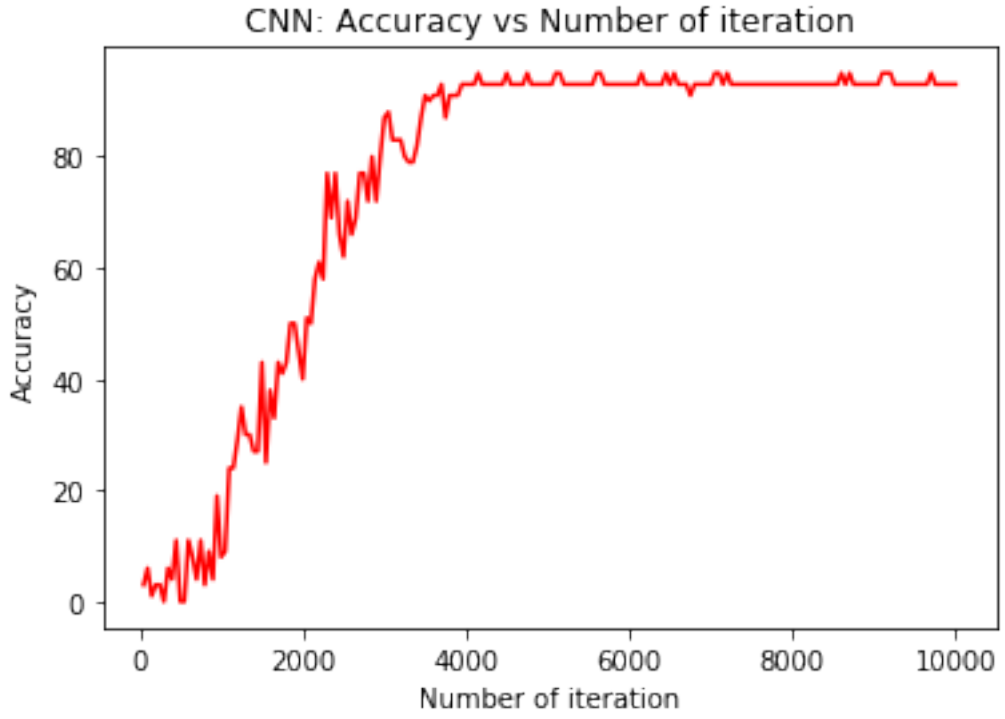


Figure 5.7: Family level Testing Accuracy

During experimentation, we train our model for different epochs, compared the results, and rerun our model on more epochs with selected parameters and the most promising validation accuracy was recorded at 40^{th} epoch using 70%, 10% and 20% training, validation and test of synthetic Geez character image dataset respectively. Once the recognition model is developed, we achieved 90 % of average recognition accuracy using the synthetic Geez character on character level and 93 % average recognition accuracy on family level.

The training loops saves the summaries in the train summary part. By using the **Tensor board** and pointing to the directory that the logs are saved, we can visualize the training procedure. For the last layer it is good to have a visualization of the distribution of the neurons outputs. By using the histogram summary the distribution of weights and biases as well as their corresponding gradients with respect to each connection weights and biases can be shown over the whole training steps. The histograms are illustrated in figures [5.10 up to 5.17].

More over ,the training loss and testing accuracy by varying the steps (iterations) and epochs can be plotted and visualized then so that the best result of the model selected.

- The initial learning rate by the Adam optimizer has been set to a small number. By setting that to a larger number, the speed of accuracy increasing could go higher. We deliberately set that to a small number to be able to track the procedure easier.

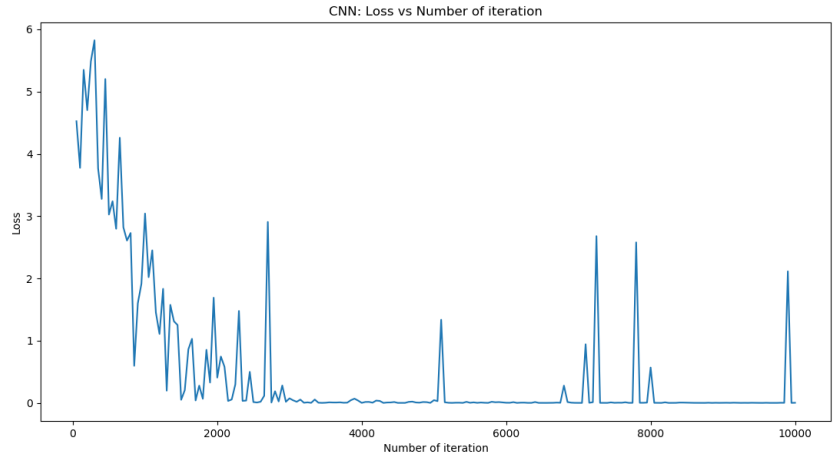


Figure 5.8: character level Training loss

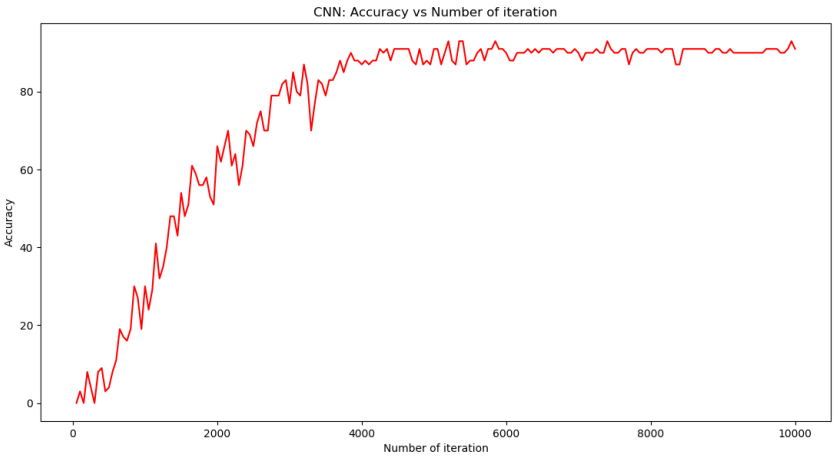


Figure 5.9: character level Testing Accuracy

- The histogram summaries are saved per each epoch and not per step. Since the generation of histogram summaries are very time-consuming, we generated per epoch of training .

5.5 PERFORMANCE COMPARISON

To the best of the researchers knowledge, there is no any publicly available dataset for Geez script recognition and all researchers have been done experiments by preparing their own dataset. Therefore, it is sometimes hard to compare, because previous work has not experimented with large database. However, in order to show the progress of Geez character image recognition, the result of the current

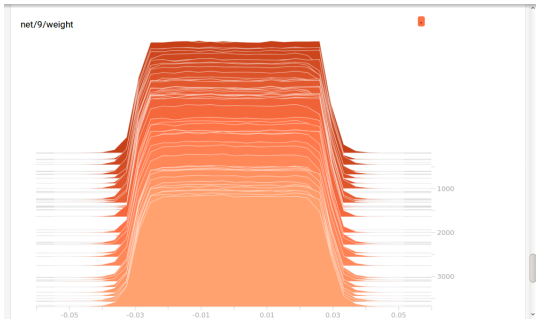


Figure 5.10: Histogram of weights for the last step (family level)

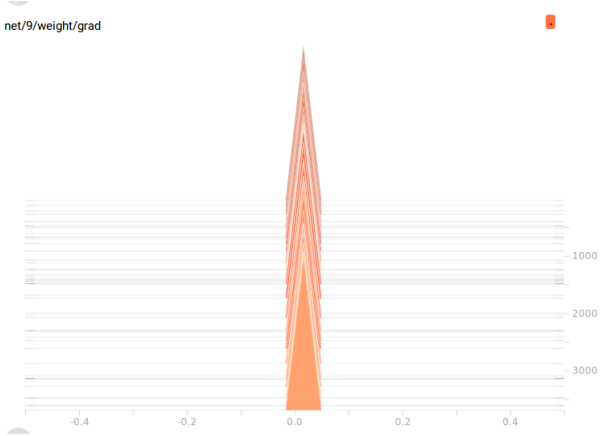


Figure 5.11: Histogram of gradient weights for the last step (family level)

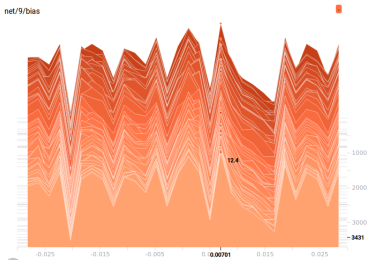


Figure 5.12: Histogram of biases for the last step(family level)

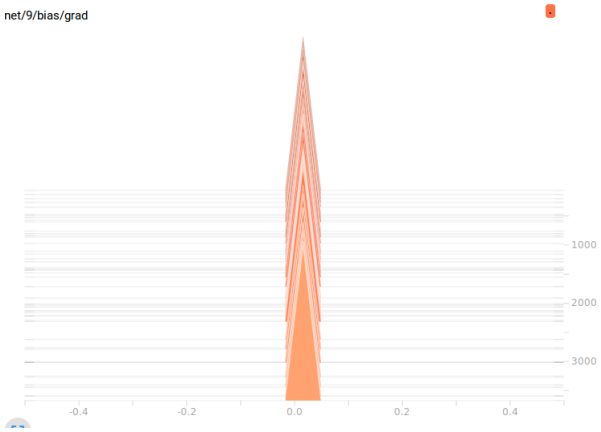


Figure 5.13: Histogram of gradient biases for the last step (family level)

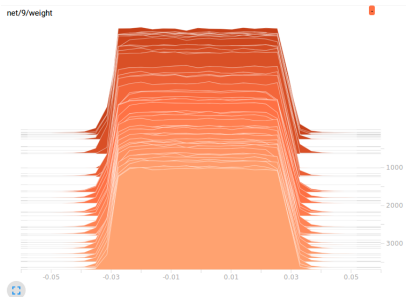


Figure 5.14: Histogram of weights for the last step (character level)

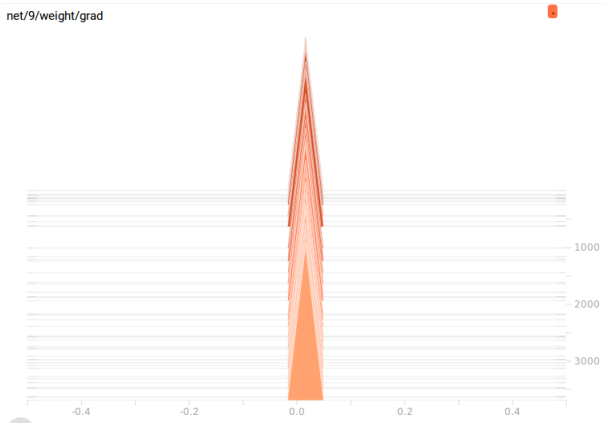


Figure 5.15: Histogram of gradient weights for the last step (character level)

Summary of Geez character		
Iteration	Training loss	Testing accuracy(%)
500	3.7004926204681396	0
2000	0.07730805873870 85	43
3500	1.108681082725525	91
5000	0.0014468358131125 57	93
6500	0.0023496006615459 92	93
8000	0.0017980964621528983	93
9500	0.002055299933999777	93
10500	0.012136055156588554	93

Table 5.1: Summary of Training process in family level

Summary of Geez character		
Iteration	Training loss	Testing accuracy(%)
500	3.02583	4
2000	0.01322	66
3500	0.00292	79
5000	0.04392	91
6500	0.00017	91
8000	0.56834	91
9500	0.00044	90
10500	0.00046	91

Table 5.2: Summary of Training process in character level

Geez character recognition model, and the results of previous researches [4], [18], [28] are shown in table 5.3

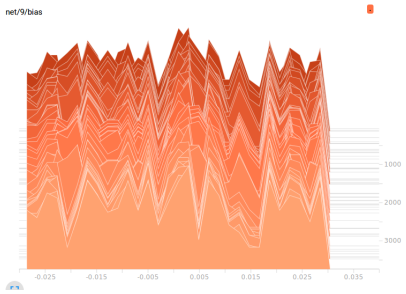


Figure 5.16: Histogram of biases for the last step (character level)

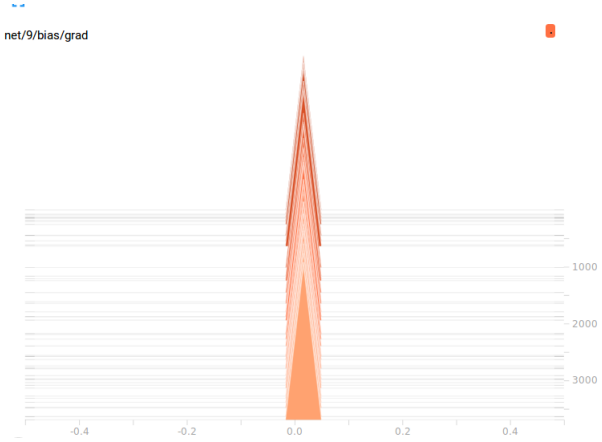


Figure 5.17: Histogram of gradient biases for the last step (character level)

Researchers	Dataset size	classfier	Accuracy(100%)
Dereje T.[4]	5172-character images	ANN	61%
Million M.[18]	7680-character images	SVM	90.37%
Yaragal A.[28]	1010-character images	ANN	73.18%
Ours	3751-character images	DCNN	91%

Table 5.3: Performance comparison of related works.

6

CONCLUSION, RECOMMENDATION AND FUTURE WORK

6.1

CONCLUSION

We experimentally developed a convolutional neural network model to recognize Geez character images. Convolutional neural network works with raw pixel image as an input rather than predefined features. We built the model considering 37 basic Geez characters family levels and 313 basic Geez characters sets i.e we use two different soft max classifiers. The experimentally developed model has 14 layers and 5,004,389 trainable parameters. Synthetic data sets are generated from OCRpy, with washra geez unicodes, in 3751 Geez character images. During training each family character exists 101 times on average. And the performance of the model is evaluated using test set which is randomly split as 20% of the total Geez character image datasets(3751),i.e.,750 character images. Our DCNN model tries to solve the Geez character recognition task through the approaches multi-class classification and multi-label classification.

After we developed and trained Geez character recognition model with the synthetically generated Geez character images, experiments were conducted using the data sets. According to the experimental results maximum recognition accuracy is at 40^{th} epoch and 10500^{th} iteration. Accordingly,we achieved an average recognition accuracy of 93% on family level and 91% on character level. In this work we have also introduced a new data set preparation technique for future Geez character recognition.

6.2

RECOMMENDATION AND FUTURE WORK

The number of data sets that we have used are very small as compared to the depth of our neural network. So my recommendation for future work to enhance the number of datasets so that the classification accuracy of the character will be improved because,nowadays there are problems in processing large data sets. Preparation, preprocessing of handwritten characters from different persons and applying to the model can also make the Geez character recognition task complete. According to our literature review luck of cooperation between the developers and researchers is also one major reason for the absence of Geez optical character recognition software. Hence, we recommend cooperation of researchers to integrate their works in order to fully digitize Geez scripts and increase datasets to train with best developed model.

As an extended research, the OCR technology towards the recognition of Geez document image can be further referenced as word and sentence level, in parallel, large corpus of historical and printed real life document images will be prepared with a benchmark experimental result.

APPENDIX

Listing A.1: Python code to generate data

```
from PIL import Image, ImageDraw, ImageFont, ImageFilter
import os
import os.path as path
#configuration
font_size=36
width=32
height=32
back_ground_color=(255,255,255)
font_size=22
font_color=(0,0,0)

"""
Data Augmentation Techniques:
1. Noise
        1.1 dots, clutter
        1.2 Scanner noise
2. Shearing, Affine Transformation
3. Different fonts
   - As MUCH AS POSSIBLE, USE ALLL OF THE GEEEZ FONTS !
"""
font_name = "Zelan"
font_ttf = "/home/tesfay/Documents/Thesis/thesis_day to day/codes/washra_fonts
    -4.1/zelan.ttf"
folder_name = font_name + "files"
os.system("mkdir " + folder_name)
lastletter = u'\u1200'
# Fist Group, 432
cntr = 0
for i in range(4116):
        print("First Round: ",i)
        #l.append(unichr(ord(lastletter) + i))
        unicode_text = unichr(ord(lastletter) + i)
        im  =  Image.new ( "RGB", (width,height), back_ground_color )
        draw  =  ImageDraw.Draw ( im )
        unicode_font = ImageFont.truetype(font_ttf, font_size)
        draw.text ( (10,10), unicode_text, font=unicode_font, fill=font_color
            )
        if True:#np.mean(im)!=255:
                cntr += 1
                file_name = path.join(folder_name, str("%05d" % cntr))
                im.save(file_name + ".jpg")
```

A.1

DATA LOADER

Listing A.2: Python code to load and transform data

```
import numpy as np
import os
import cv2 as cv
import argparse


"./imgs"
# Load the twor rows
dataset = np.loadtxt('labels.txt', comments='#', usecols = (1,2))
names = np.loadtxt('labels.txt', comments='#', usecols = (0))
n = []
for n_i in names:
        t = str(int(n_i)).zfill(5)
        n.append(t)
names = n


labels = dataset[:, 0]
family = dataset[:, 1]

imgs = []
filenames = sorted(os.listdir("./imgs"))
for name in names:
        img_name = os.path.join("./imgs", name + '.jpg')
        imgs.append(cv.imread(img_name, 0))



# Now lets save the images and the labels into a file
trainX = np.stack(imgs)
trainY1 = np.stack(labels)
trainY2 = np.stack(family)
os.makedirs('./dataset/', exist_ok=True)
np.save('./dataset/' + '_train_imgs.npy', trainX)
np.save('./dataset/' + '_train_labels.npy', trainY1)
np.save('./dataset/' + '_train_family.npy', trainY2)

# Print the shape of the training images, it should be [B,32,32,3]
# B in this case is eq to 14
print(trainX.shape)
print(trainY1.shape)
```

A.2

DATA LABEL READER

Listing A.3: Python code to load data labels

```
import os
import csv
fid = open("labels.txt", "r")
reader = csv.reader(fid)
data = reader.read()
fid.close()
print(data)
```

BIBLIOGRAPHY

[1] Adnan Ul-Hasan. "Generic text recognition using long short-term memory networks." In: (2016).

[2] Nikolai Gorski, Valery Anisimov, Emmanuel Augustin, Olivier Baret, David Price, and J-C Simon. "A2ia check reader: A family of bank check recognition systems." In: *icdar*. IEEE. 1999, p. 523.

[3] Michal Bušta, Tomáš Drtina, David Helekal, Lukáš Neumann, and Jiří Matas. "Efficient character skew rectification in scene text images." In: *Asian Conference on Computer Vision*. Springer. 2014, pp. 134–146.

[4] Dereje Teferi. "Optical Character Recognition of Typewritten Amharic Text." PhD thesis. Masters thesis) School of Information studies for Africa, Addis Ababa . . ., 1999.

[5] Henry S Baird and Karl Tombre. "The evolution of document image analysis." In: *Handbook of document image processing and recognition* (2014), pp. 63–71.

[6] Jinfeng Bai, Zhineng Chen, Bailan Feng, and Bo Xu. "Image character recognition using deep convolutional neural network learned from different languages." In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, pp. 2560–2564.

[7] John Makhoul, Richard Schwartz, Christopher Lapre, and Issam Bazzi. "A script-independent methodology for optical character recognition." In: *Pattern Recognition* 31.9 (1998), pp. 1285–1294.

[8] Mohamed Saad Mostafa El-Mahallawy. "A large scale HMM-based omni front-written OCR system for cursive scripts." In: *Cairo University, Faculty of Engineering* (2008).

[9] Sheikh Faisal Rashid. "Optical Character Recognition-A Combined ANN And HMM Approach." In: (2014).

[10] Alex Graves. "Supervised sequence labelling." In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.

[11] Vijay Chandrasekhar, Gabriel Takacs, David Chen, Sam Tsai, Radek Grzeszczuk, and Bernd Girod. "CHoG: Compressed histogram of gradients a low bit-rate feature descriptor." In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 2504–2511.

[12] Anna Babaryka. *Recognition from collections of local features*. 2012.

[13] James Hafner, Harpreet S. Sawhney, William Equitz, Myron Flickner, and Wayne Niblack. "Efficient color histogram indexing for quadratic form distance functions." In: *IEEE transactions on pattern analysis and machine intelligence* 17.7 (1995), pp. 729–736.

[14] Yaregal Assabie and Josef Bigun. "Structural and syntactic techniques for recognition of Ethiopic characters." In: *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer. 2006, pp. 118–126.

[15] Yaregal Assabie and Josef Bigun. "Ethiopic character recognition using direction field tensor." In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 3. IEEE. 2006, pp. 284–287.

[16] Fitsum Demissie. *Developing Optical Character Recoginition for Ethiopic Scripts*. 2011.

[17] Million Meshesha and CV Jawahar. "Recognition of printed Amharic documents." In: *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE. 2005, pp. 784–788.

[18] Million Meshesha and CV Jawahar. "Optical character recognition of Amharic documents." In: *African Journal of Information & Communication Technology* 3.2 (2007).

[19] Halefom Tekle Weldegebriel, JinXiu Chen, and Defu Zhang. "Deep learning for Ethiopian Ge'ez script optical character recognision." In: *Advanced Computational Intelligence (ICACI), 2018 Tenth International Conference on*. IEEE. 2018, pp. 540–545.

[20] Abay Teshager Birhanu and R Sethuraman. "Artificial Neural Network Approach to the Development of OCR for Real Life Amharic Documents." In: *International Journal of Science, Engineering and Technology Research* 4.1 (2015), pp. 141–147.

[21] Million Meshesha. "Recognition and retrieval from document image collections." PhD thesis. Ph. D. Thesis, IIIT Hyderabad, India, 2008.

[22] Lalith Premaratne, Yaregal Assabie, and Josef Bigun. "Recognition of modification-based scripts using direction tensors." In: *4th Indian Conference on Computer Vision, Graphics and Image, December 16-18, 2004, Kolkata, India*. 2004, pp. 587–592.

[23] ABAY TESHAGER BIRHANU. "SCHOOL OF GRADUATE STUDIES FACULTY OF INFORMATICS DEPARTMENT OF INFORMATION SCIENCE." PhD thesis. Addis Ababa University, 2010.

[24] Yaregal Assabie and Josef Bigun. "Lexicon-based offline recognition of Amharic words in unconstrained handwritten text." In: *2008 19th International Conference on Pattern Recognition*. IEEE. 2008, pp. 1–4.

[25] Yaregal Assabie and Josef Bigun. "Online Handwriting Recognition of Ethiopic Script." In: *Eleventh International Conference on Frontiers in Handwriting Recognition (ICFHR2008)*. 2007, pp. 153–158.

[26] Ahmed El-Sawy, Mohamed Loey, and EB Hazem. "Arabic handwritten characters recognition using convolutional neural network." In: *WSEAS Transactions on Computer Research* 5 (2017), pp. 11–19.

[27] Siranesh Getu Endalamaw. "Ancient Ethiopic Manuscript Recognition Using Deep Learning Artificial Neural Network." PhD thesis. Addis Ababa University, 2016.

[28] Yaregal Assabie. "Optical character recognition of Amharic text: an integrated approach." PhD thesis. Master's thesis, School of Information Studies for Africa, Addis Ababa . . ., 2002.

[29] Maad Shatnawi and Sherief Abdallah. "Improving handwritten arabic character recognition by modeling human handwriting distortions." In: *ACM Transactions on Asian and Low-Resource Language Information Processing* 15.1 (2016), p. 3.

[30] Haifeng Zhao, Yong Hu, and Jinxia Zhang. "Character recognition via a compact convolutional neural network." In: *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE. 2017, pp. 1–6.

[31] Rainer Voigt. "The gemination of the present-imperfect forms in Old Ethiopic." In: *Journal of Semitic Studies* 35.1 (1990), pp. 1–18.

[32] K Pramod Sankar, Vamshi Ambati, Lakshmi Pratha, and CV Jawahar. "Digitizing a million books: Challenges for document analysis." In: *International Workshop on Document Analysis Systems*. Springer. 2006, pp. 425–436.

[33] Lars Asker, Atelach Alemu Argaw, Björn Gambäck, Samuel Eyassu Asfeha, and Lemma Nigussie Habte. "Classifying Amharic webnews." In: *Information retrieval* 12.3 (2009), pp. 416–435.

[34] Vishal Maini and Samer Sabri. *Machine Learning for Humans*. 2017.

[35] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural networks* 61 (2015), pp. 85–117.

[36] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. "Deep convolutional ranking for multilabel image annotation." In: *arXiv preprint arXiv:1312.4894* (2013).

[37] karpathy. *Convolutional Neural Networks (CNNs / ConvNets)*. http://cs231n.github.io/convolutional-networks/. [Online; accessed 19-July-2015]. 2014.

[38] Thomas M Breuel. "The OCRopus open source OCR system." In: *Document Recognition and Retrieval XV*. Vol. 6815. International Society for Optics and Photonics. 2008, 68150F.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[40] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. "Deeply-supervised nets." In: *Artificial Intelligence and Statistics*. 2015, pp. 562–570.

[41] Li Chen, Song Wang, Wei Fan, Jun Sun, and Naoi Satoshi. "Reconstruction combined training for convolutional neural networks on character recognition." In: *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE. 2015, pp. 431–435.

[42] Yann LeCun et al. “Generalization and network design strategies.” In: *Connectionism in perspective*. Vol. 19. Citeseer, 1989.