A Structure-preserving Document Conversion System for Manipuri Documents in Bengali Script to Meetei Script

Thesis submitted in partial fulfilment of the requirements
for the award of the degree of

Doctor of Philosophy

in

Centre for Linguistic Science and Technology

by

Jennil Thiyam

Under the supervision of

Dr. Sanasam Ranbir Singh and Prof. Prabin Kumar Bora

Centre for Linguistic Science and Technology
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

June, 2023

Dedicated to

My beloved Parents,

and

My dear Brothers and Sisters

Acknowledgment

I am deeply grateful to my esteemed supervisors, Dr. Sanasam Ranbir Singh and Prof. Prabin Kumar Bora, for their unwavering support, patience, and invaluable guidance throughout my doctoral research journey. Their constant encouragement and positive mentorship have been instrumental in shaping me both as a researcher and an individual. I am truly indebted to them for the opportunity to work under their guidance. I would like to extend my heartfelt appreciation to the members of my thesis doctoral committee - Prof. Priyankoo Sarmah, Dr. Samit Bhattacharya, and Prof. Sukumar Nandi, for their insightful comments and suggestions, which have significantly improved the quality and clarity of my work. I would also like to acknowledge the heads of the Centre for Linguistic Science and Technology (CLST) at IITG, Prof. Sukumar Nandi and Prof. Rohit Sinha, for providing me with the necessary facilities and resources to conduct my research. I am grateful to the staff of CLST, especially Mr. Souvik Chowdhury and Mr. Raktajit Pathak, for their unwavering support in addressing any engineering and administrative matters that arose during my doctoral journey. I would like to express my heartfelt appreciation to all my colleagues and friends who have accompanied me during my Ph.D. years. I am particularly grateful to my fellow lab mates at CLST and OSINT, especially Bornali, Shikha, Seema, Hemanta, Pankaj, Gyanendro, and Sujit, for creating a stimulating and collaborative environment. The engaging discussions, collaborative problem-solving, and teamwork have had a profound impact on my growth as an independent and motivated researcher.

My deepest appreciation goes to a set of people who have been my pillars of strength throughout my Ph.D. journey. First of all, My parents, Mrs. Th. Motibala Devi and Mr. Thiyam Basanta Kumar, despite our financial challenges, have been the driving force behind my pursuit of a Ph.D., promising unwavering support. My elder brother, Mr. Jupiter Thiyam, and younger brother, Mr. Kiyamba Thiyam, have stood by me through thick and thin. My elder sister, Ingudam Prishila Devi, and younger sister, Ingudam Joypriya Chanu, have provided constant love and inspiration. My dear one, Pheiroijam Prishika, a cherished companion, has infused my life with joy, support, and motivation. Among them all, my grandmother, Thokchom Nungshi Devi, holds a special place. Her wisdom, love, and valuable life lessons have been instrumental in shaping who I am today. She is, without doubt, my favorite person. My grandfather, Thokchom Mani Singh, has also offered invaluable wisdom and guidance. This set of my pillars would not be complete without my main supervisor, Dr. Sanasam Ranbir Singh. No words can adequately express the gratitude I have for him. He is an exceptional mentor and the guiding force behind my Ph.D. journey. I consider myself incredibly fortunate to have had the opportunity to work under his guidance. He has been more than a supervisor to me; he has been a source of inspiration, support, and encouragement. Throughout my journey, Dr. Singh has been there during both the rainy and cloudy days, brightening them with his guidance and mentorship. He has provided me with the

tools, resources, and opportunities to thrive in my research endeavors. His insightful feedback, constructive criticism, and attention to detail have pushed me to constantly improve and strive for excellence. Beyond his exceptional guidance, Dr. Singh has shown genuine care and concern for my overall well-being. He has created a nurturing and supportive environment that fosters personal and professional growth. I am grateful for his patience, understanding, and willingness to go above and beyond to help me overcome obstacles and challenges. From the bottom of my heart, I extend my sincerest wishes to him for continued success, good health, and fulfillment in all his endeavors. I am forever indebted to him for his unwavering support and the positive impact he has had on my life and career. Thank you, Dr. Singh, for being an exceptional supervisor and mentor. Working under your guidance has been a privilege, and I am eternally grateful.

Finally, I would like to express my sincere gratitude to IIT Guwahati for providing me with a remarkable academic experience. The institution's commitment to excellence in education and research has played a vital role in shaping my intellectual growth. Thank you for the invaluable opportunities and experiences I have gained during my time at IIT Guwahati. I am proud to be associated with such an esteemed institution.

January 13, 2023 Jennil Thiyam

Declaration

I certify that

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisors.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.
- I am fully aware that my thesis supervisors are not in a position to check for any possible instance of plagiarism within this submitted work.

January 13, 2023 Jennil Thiyam

Centre for Linguistic Science and Technology
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

Certificate

This is to certify that this thesis entitled "A Structure-preserving Document Conversion System for Manipuri Documents in Bengali Script to Meetei Script" submitted by Jennil Thiyam, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Assam, India, is a record of the bonafide research work carried out by him under my guidance and supervision at the Centre for Linguistic Science and Technology, Indian Institute of Technology Guwahati, Assam, India. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date: January 13, 2023
Place: Guwahati

Dr. Sanasam Ranbir Singh
(Coordinating supervisor)
Associate Professor
Dept. of C.S.E
IIT Guwahati

Prof. Prabin Kumar Bora
(Co-supervisor)
Professor
Dept. of E.E.E
IIT Guwahati

A Structure-preserving Document Conversion System for Manipuri Documents in Bengali Script to Meetei Script

Abstract

Manipuri, or Meeteilon, is one of the resource-poor languages of India and the lingua franca of the Indian state of Manipur. Though the *Meetei Mayek* (Manipuri script) is known to use for writing Manipuri documents since the early 6^{th} AD, it was banned and replaced with the *Bengali script* by the then king of Manipur in the 18^{th} century. Since the late 70s, the Government of Manipur has made an effort to reintroduce Meetei Mayek and included it in Unicode in the year 2009. Meetel Mayek is progressively replacing the Bengali script in schools, colleges, offices, and other places. During the era of using Bengali script as Manipuri writing script (more than 300 years), a huge volume of Manipuri documents has been created in Bengali script. Almost all of the Manipuri literary materials are in Bengali script, and the population in Manipur is broadly divided into - *Bengali script literate* and *Meetei Mayek literate*. After a few decades, the majority of the Manipuri population will not be able to read/write Bengali script, creating a huge gap in accessing literary materials. Therefore, there is an urgent need to develop an effective system to convert Manipuri documents written in the Bengali script to Meetei Mayek to bridge the script divide.

Motivated by the above concern, this thesis focuses on the following three research problems associated with the development of an automatic document conversion system (DCS) for the Manipuri documents in the Bengali script to Meetei Mayek.

- **Document segmentation and region classification system:** Document segmentation and region classification are the first and foremost tasks in developing a DCS. In document region segmentation and classification, one of the prominent challenges is effectively segmenting non-textual regions that contain sparsely clustered pixels. While previous studies have primarily concentrated on using a single model to segment regions of interest (textual or non-textual), this thesis proposes a novel 2-tier feedback-based end-to-end deep learning and rule-based integrated framework. The framework aims to address this challenge by enabling joint segmentation and identification of regions of interest in a more efficient and effective manner. In addition, a dataset (document images and their corresponding mask images) for future similar research activities has also been created.

- **Chart type classification model:** In the field of chart type classification, false classification poses a significant challenge due to two main factors. First, there are confusing chart type pairs where multiple chart types exhibit very similar

characteristics. Second, noisy samples significantly contribute to misclassification as charts often contain additional components such as textual elements, the information presented in shapes, and marking points. To tackle these challenges, this study proposes a novel approach based on an attention and triplet loss-based Convolutional Neural Network (CNN) framework which enhances the ability of the model to distinguish between similar chart types and mitigate the impact of noisy samples. In addition, a dataset of 28 chart types has been proposed for future similar research.

- **Manipuri OCR system:** The development of an Optical Character Recognition (OCR) system from scratch poses a challenge, particularly when it comes to low-resource languages. Obtaining large text corpora from diverse environments is crucial for the effective functioning of the OCR system. However, curating such extensive corpora proves to be a challenging task for low-resource languages, as the availability of digitized textual data is severely limited. To address this challenge, this thesis employs an adaptive approach by leveraging existing OCR systems designed for similar scripts. By adopting this adaptive approach, the thesis aims to overcome the scarcity of resources for the low-resource language, Manipuri.

A prototype of the proposed DCS is implemented, and details are presented in Appendix A. In summary, this thesis makes important research contributions in terms of datasets and models for document segmentation, chart classification, and optical character recognition and develops a prototype end-to-end DCS.

Contents

Listing of figures

List of Tables

List of Abbreviations

Terms	Abbreviations
AI	Artificial Intelligence
BN	Batch Normalization
CBAM	Convolutional Block Attention Module
CCOMW	Conjunct Characters Oriented Manipuri Words
CLA	Character Level Accuracy
CNN	Convolutional Neural Network
CER	Xception-Middle
DL	Deep Learning
DCS	Document Conversion System
DNN	Deep Neural Network
ETSM	Equation and Textual Segmentation Model
ETNC	Equation Textual and Noise Classifier
FC	Fully-connected
GOBD	Graphical Object-Blank dataset
GLCM	Gray Level Co-Occurrence Matri
GSM	Graphical-region Segmentation Model
GC	Graghical-region Classifier
HOG	Histogram of Oriented Gradients
IoU	Intersection over Union
IH	In-house
IH-A	In-House-All
ICOMW	Individual Characters Oriented Manipuri Words
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbor
LRN	Local Response Normalization
LSTM	Long Short-Term Memory
LBP	Local Binary Pattern
MLP	Multi Layer Perceptron
ML	Machine Learning
MRW	Manipuri Random Words
NB	Naive Bayes
OCR	Optical Character Recognition

PDF	Portable Document Format
RF	Random forest
ReLu	Rectified Linear Uni
RNN	Recurrent Neural Network
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine
SE	Squeeze and Excitation
SGD	Stochastic Gradient Descent
TEND	Text-Equation-Noise dataset
TNMC	Total noise misclassification
TNMCO	Total noise misclassification overall
TCMC	Total Confusing pairs misclassification
TCMCO	Total Confusing pairs misclassification overall
TCBAM-XMEX	Triplet loss based CBAM-XMEX
TCBAM-X	Triplet loss based CBAM-Xception*
XEN	Xception-Entr
XM	Xception-Middle
XEX	Xception-Exit
XMEX	Xception-Middle-Exit
XA	Xception-All

"Life isn't about black and white,look around and you will see that the world is much more colorful than you thought"

Tal bekerman

1 Introduction

Manipuri, or Meetei, is one of the 22 official languages of India[87]. It is widely spoken in the Indian state of Manipur. It is also spoken by small communities in two other states of India: Assam and Tripura, and also in two neighboring countries: Bangladesh and Myanmar. Manipuri is written generally using two scripts: Bengali script and Meetei Mayek(Mayek is the synonym for script in the Meetei language and they are used interchangeably in this thesis). Meetei script was used to write the Meetei language from 33 AD until the early 18th century[93]. In the year 1717, under the command of the then

ruler of Manipur (an independent kingdom till 1949), the Meetei script was replaced with the Bengali script for writing Manipuri documents. Even though researchers around the state had introduced a modernized version of the ancient Meetei script in 1980, it was only in 2009 that the script was encoded in Unicode. Since then, the Meetei script has been used in various fields, such as schools, offices, and businesses, along with the Bengali script. It is likely that after a few years, the original script will be used as a primary script for writing Manipuri documents. However, over the last 300 years, a huge volume of Manipuri documents in Bengali script has been created. Almost all of the Manipuri literary materials are in Bengali script, and the population in Manipur is broadly divided into - *Bengali script literate* and *Meetei Mayek literate*. Considering the new development in regards to the Manipuri writing system, after a few decades, the majority of the Manipuri population will not be able to read/write Bengali script, creating a huge gap in accessing literary materials. Therefore, there is an urgent need to develop an effective system to convert Manipuri documents written in Bengali script to Meetei script without distorting the structural properties of the documents. Motivated by the urgent need to develop an automatic system for converting a Manipuri document (in image form) written in Bengali script to Meetei script, this thesis focuses on the research to develop core technologies of various modules which are required for building a DCS (document conversion system) without distorting structural properties. To the best of our knowledge, this study is the first such effort to develop an end-to-end document conversion system.

1.1 An Overview of Document Conversion System

To align the research activities reported in this thesis from the perspective of a document conversion system, a typical DCS has been conceptualized and implemented. The conceptualized DCS is briefly discussed first. Given a document image, it has two expected

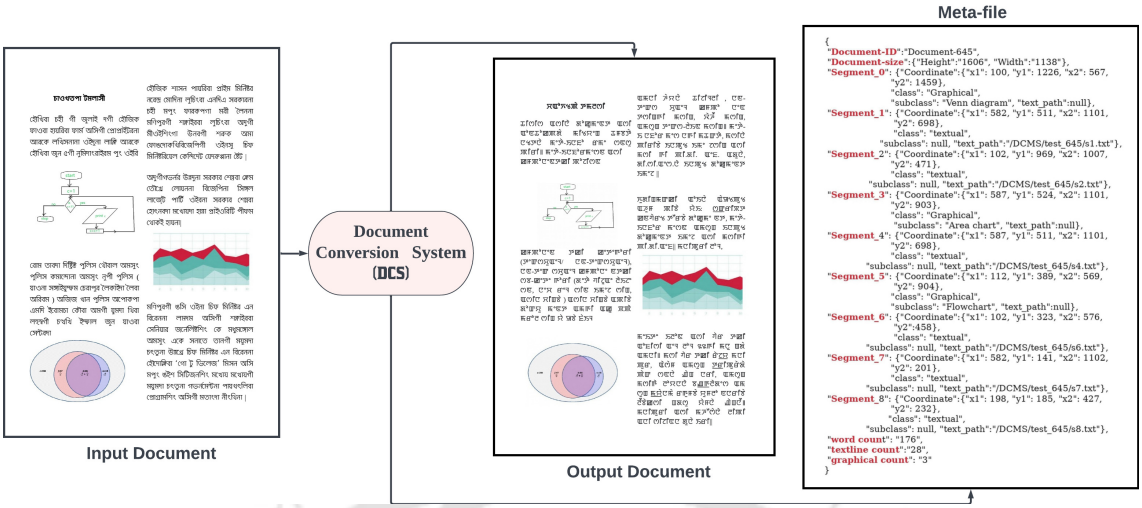


Figure 1.1: Expected outputs of DCS for any given Manipuri document images written in Bengali script.

outputs as illustrated in Figure 1.1 - *a document in the target script maintaining the structural properties of the input document* and *a meta-file*. The meta-file contains information related to (i) *document structures*: document size, and positional information of segments or regions (textual/non-textual), (ii) *document contents (textual and non-textual)*: textual and non-textual contents in the document are saved as the individual segmented images, and (iii) *content description*: descriptive information of the contents, such as number of words, number of text lines, number of graphical components, content types, image classes (like a chart, equation, table, medical image, natural scene image, and sketches). A typical schematic workflow diagram of the DCS is shown in Figure 1.2. It consists of six broad modules, as briefly explained below.

- *Region segmentation and classification:* This module is expected to segments the input document image into multiple regions of interest and classifies them into three classes: *textual*, *equation*, and *graphical*. In other words, its task is to find the regions of interest in the document image. The components considered under the class *graphical*, may differ among various studies. This study considers all other

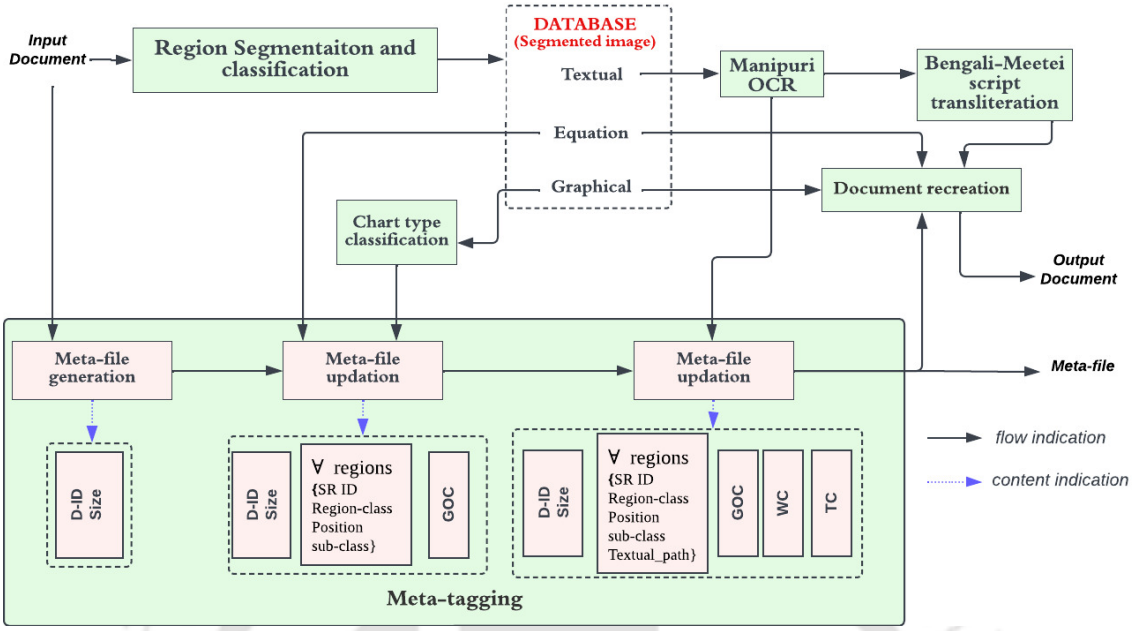


Figure 1.2: Schematic workflow of DCS. D-ID denotes the document identification (ID), Size indicates the size of the input document image, SR ID denotes the segmentation region ID (which the system gives to every segmented region). Region Class presents the class of the region (Textual/Equation/Graphics), Position is the coordinates of the bounding box of the segmented regions, and sub-class gives the sub-class of the regions (28 chart types and "Other"). It is NULL for Region Class = (*Textual* || *Equation*). Textual_path gives the path to a text file which contained textual components in its editable form. It is NULL for Region Class= (*Equation* || *Graphical*). GOC, WC and TC denote Graphical Object Count, Word Count, and Text-line Count in the given input document.

components, other than the text and equation, such as a chart plot, table, medical image, natural scene image, and sketches, as graphics. As shown in Figure 1.2, DCS is expected to save the segmented and classified regions as individual images in a DATABASE.

- *chart type classification:* This module is expected to identify the types of graphical components present in the document. Identification of the types of graphical components present in a document is one of the important research areas in document analysis. While areas such as object recognition[19], face recognition[20], and medical image processing[46] have been extensively studied, chart type classification (an

important task in document analysis) is relatively under-explored. Motivated by this, the thesis focuses on proposing an effective chart classification method to identify various types of scientific charts. At the same time, the identification of other graphical objects may be adopted from existing literature or tools.

- *Manipuri OCR (Optical Character Recognition):* This module is expected to convert the Manipuri-Bengali textual components into their editable form.
- *Bengali-Meetei script transliteration:* The objective of this module is to convert editable Manipuri text in Bengali script to the Meetei script.
- *Meta-tagging:* This module will generate a meta-file for each input document image. It can be seen from Figure 1.2, as the input image passes through the modules, the content of the meta-file will be updated. Finally, an updated meta-file will be generated for the output document image.
- *Document recreation:* This module will collect information from other modules, namely, *Bengali-Meetei script transliteration*, *meta-tagging*, and segmented images (of equations and graphical components) from the *DATABASE* to recreate the input document image as a PDF file. The recreated document will preserve all the information, including structural information.

Among the above six modules, the proposed research works reported in this thesis mainly focus on the first three modules, namely, *region segmentation and classification, chart type classification* and *Manipuri OCR*.

1.2 Challenges

While developing the three modules of *region segmentation and classification, chart type classification* and *Manipuri OCR*, this work encounters various challenges as described

below:

- **Segmentation:** The module *Region segmentation and classification* has two major tasks: the segmentation of the regions of interest in the given input documents and the classification of the segmented/predicted regions into three classes: *textual, equation,* and *graphical*. Among these two tasks, the major challenge comes with the segmentation of the regions of interest. The difference in the characteristics or structures of the above three classes leads to false segmentation. Because of the various fonts, styles, and sizes in which they can be represented, the segmentation of textual components and equation components are limited to the specific fonts, styles, and sizes. This limitation becomes prominent when the segmentation model is based on conventional machine learning methods where features are manually extracted. However, with the evolution and advancement in the field of deep learning, the challenges of explicit feature engineering are reduced. However, the diverse nature of possible documents, such as newspapers, journals, books, advertisements, and magazines, limits the deep-learning-based segmentation models to a specific type of document in most cases. As this study focuses on the documents of book or journal types, the segmentation of text and equation becomes less problematic because of the uniform standard fonts, styles, and sizes used. The primary challenge is the accurate segmentation of graphical components. Based on the distribution of pixels in the regions, graphical components could be broadly divided into two categories: (i) *graphical regions with sparsely clustered pixels:* most of the foreground pixels of a graphical region are connected by only 4-neighbor (horizontal/vertical) connectivity∗, (ii) *graphical regions with tightly clustered pixels:* contrary to the prior, most of the foreground

∗A pixel p at (x, y) has 4-neighbour (horizontal/vertical) at $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ and $(x, y-1)$.

pixels of a graphical region are connected by 4-neighbor or diagonal-neighbor∗, or 8-neighbor connectivity†. The accurate segmentation of graphical regions with sparsely clustered pixels is one of the challenges because of the nature of the pixel distribution.

- **Classification:** The module chart type classification belongs to one of the under-researched domains. To date, there have been fewer than one hundred studies on chart type classification. The main reason is the unavailability of chart images and inconsistency in the number of chart types. From the various state-of-the-art literature, this study observes four main characteristics: (i) inconsistencies in the performance of various chart type classification methods, (ii) inconsistencies in the number of chart types considered, (iii) inconsistencies in the size of the dataset, and (iv) misclassification because of two errors: *Chart's noise:* Chart samples with noise, such as a chart sample with textual content covering more area than the plot, chart samples with hard background grids, and chart samples with numerous textual components, and *Confusing chart type pairs:* Chart samples, which are classified as another chart type because of their highly similar characteristics, e.g., Venn diagrams and pie charts, scatter charts and node links.

- **Manipuri OCR:** Even though OCR has more than 50 years of history, the research on the development of OCR for the Manipuri language is in its infancy stage because of the lack of a resource (textual images and their corresponding transcripts). As the Manipuri-Bengali script is similar to the Bengali and Assamese scripts (with little differences), this study performed a brief survey on the development of OCR of these two scripts. Motivated by the similarities, this study provides a brief survey on the OCR development of these two scripts. One

∗A pixel p at (x, y) has 4 diagonal neighbors at $(x+1, y+1), (x+1, y-1), (x-1, y+1)$ and $(x-1, y-1)$.

†The combination of 4-neighbor and the diagonal - neighbor of pixel p.

common challenge observed in the development of OCR of these two scripts is the complex structures of the letters, mainly the conjunct characters (the characters made up with the combination of multiple characters such as পব (combination of প and ব), স্প (combination of স and প)). This problem is a big threat to OCR development using rule-based or traditional machine learning methods. However, with an advanced learning method such as deep learning, it becomes less of a problem if training text samples are abundant. Various studies began to develop Assamese OCR[9] and Bengali OCR[101] using deep learning methods that work on data sequences such as RNN[88], LSTM[156], and so on. In particular, an open-source framework, Tesseract[128] developed the effective OCR systems for these two scripts with LSTM as the backbone. The systems were trained on the extensive size of the respective text corpora. Tesseract has provided a framework to develop OCR for any script or language. Despite the fact that a framework is readily available, Manipur-Bengali being one of the poor-resourced scripts or languages, there is no significant training text. Therefore, the unavailability of the digitized Manipur-Bengali text becomes one of the major challenges.

1.3 Contributions

This thesis aims to address the challenges discussed above. The contribution made in the thesis can be broadly divided into two: (i) the proposed model and (ii) the dataset.

1. **Proposed models:** In this thesis, three models or systems are proposed:

   (a) *Document segmentation and region classification system:* In this contribution, a document segmentation framework that labels the regions of interest in three classes— *textual*, *equation*, and *graphical*, is proposed. The proposed framework is able to address the major issues of existing studies of inaccurate

segmentation of specific graphical components (graphical regions that have minimal foreground pixels compared to the background pixels), which was discussed earlier in Section 1.2.

(b) *chart type classification model:* This thesis proposes an attention-triplet loss-based chart type classification. It addresses the misclassification caused by the chart's noise and confusing chart-pairs, which was discussed earlier in Section 1.2 .

(c) *Manipuri OCR system:* This thesis proposes a Manipuri-Bengali OCR system. To overcome the major challenge that we have discussed earlier in Section 1.2, which is the lack of Bengali scripted Manipuri textual corpus, the OCR system is developed with the approach of adapting existing OCR of Bengali and Assamese scripts.

2. **Datasets:** In this thesis, three datasets are curated: a text-based dataset and two image-based datasets:

(a) *Document images:* This dataset consists of 4840 document images with their corresponding annotated masks. The textual components in the document images are written in Latin (3840) and Bengali (1000) scripts. All of them consist of graphical components/regions. This dataset is used in the development of a document segmentation model, which is presented in Chapter 3.

(b) *chart types:* This dataset consists of 117,271 chart images over 28 chart types. They are collected from the response of browser Google and research papers published in six different journals/conferences. This dataset is the largest chart type dataset in terms of the number of chart types and the number of samples. This dataset is used in the empirical analysis of the chart type

classification presented in Chapter 4 and the development of a chart type classification model, which is presented in Chapter 5.

(c) *Manipuri-Bengali text corpus:* A Manipuri-Bengali text corpus of 72,634 words is curated. It has 7300 (approx.) sentences with 20 words in average. They are collected from a Manipur news daily. This dataset is used in the development of Manipuri OCR, which is presented in Chapter 6.

1.4 Organization of the Thesis

The rest of the thesis are organized as follows:

- **Chapter 2 Background Studies:** This chapter presents the discussion over the domains which are exploited by this thesis work, namely convolutional neural network, image segmentation model UNet, different types of attention models in image processing, transfer learning, and an OCR system.
- **Chapter 3 Integrated Document Segmentation and Region Identification- Textual, Equation, and Graphical:** This chapter presents two contributions: a proposed model *Document Segmentation and Region Classification System*, and a document image dataset *Document images* .
- **Chapter 4 chart type Classification: Empirical Analysis:** This chapter presents two contributions- an *Empirical analysis of chart type classification*, and a chart type dataset *chart types* . The analysis is done over 35 existing ML (machine learning)-based image classification frameworks, including various state-of-the-art chart type classification models. The discovery of two specific errors (chart's noise and confusing chart-pairs) encountered by various chart type classifications is discussed here.

- **Chapter 5 chart type Classification: An Attention-triplet-loss based chart type Classification:** This chapter presents the proposed chart type classification model that addresses the issues introduced by two specific errors, namely, chart noise and confusing chart-pairs.

- **Chapter 6 Manipuri OCR :** This chapter presents two contributions - a Manipuri Bengali OCR system and a Bengali scripted Manipuri text corpus, *Manipuri-Bengali text corpus*. It first presents the performance study of two OCRs for similar scripts (Assamese and Bengali) on Bengali-scripted Manipuri documents. Based on the study, the Manipuri Bengali OCR system is developed.

- **Chapter 7 Conclusion and Future Work:** This chapter concludes the thesis with a summary of the work done and mentions possible future research directions.

By integrating the proposed models described above in chapter 3, chapter 5, and chapter 6, and three other modules that we have mentioned earlier, namely *Bengali-Meetei script transliteration system, Meta-tagging*, and *Document recreation* which are developed with existing technologies, the conceptualized DCS has been implemented, and its implementation details are provided in Appendix A.

"A man's feet should be planted in his country,
but his eyes should survey the world"

George Santayana

2

Background Studies

This chapter provides a concise overview of the various fields that have been utilized in this thesis, including document layout analysis, machine learning-based image classification, OCR, and machine transliteration. These fields serve as the foundation for the proposed methods presented in this work, such as the document segmentation and classification model, the chart-type classification model, and the Manipuri OCR adapted from the existing Bengali OCR system. The utilization of image classification methods, the UNet image segmentation model, transfer learning with pretrained weights,

attention mechanisms, and the Tesseract platform for OCR development are among the techniques employed in this research. The background studies in this chapter help establish the necessary groundwork for the subsequent chapters, showcasing the application of these techniques in practice.

2.1 Image Classification

Image classification is a fundamental task in which a label is assigned to an entire image. Typically, each image is associated with only one class. Image classification models take an image as input and provide a prediction regarding which class the image belongs to. Over time, with the advancement of machine learning (ML)[110], image classification approaches can be broadly categorized into two main groups: (i) conventional image classification methods and (ii) deep learning image classification methods. The workflow of these two approaches is depicted in Figure 2.1. In the case of the conventional approach, there is a manual feature engineering step involved. This step requires domain expertise to carefully design and extract meaningful features from the images. These handcrafted features are then used as input to a machine learning algorithm, such as support vector machines (SVM)[56] or random forests (RF)[11], to perform the classification task. On the other hand, deep learning-based image classification methods have gained significant popularity in recent years. These methods leverage deep neural networks, specifically designed for image analysis, to automatically extract relevant features from the input images. The process of feature extraction is done implicitly by the deep network through multiple layers of interconnected neurons. The network learns to recognize intricate patterns and representations within the images, enabling it to make accurate predictions. Deep learning models, such as convolutional neural networks (CNNs)[76], have demonstrated remarkable performance in image classification tasks, often surpassing the accuracy achieved by conventional approaches. They are

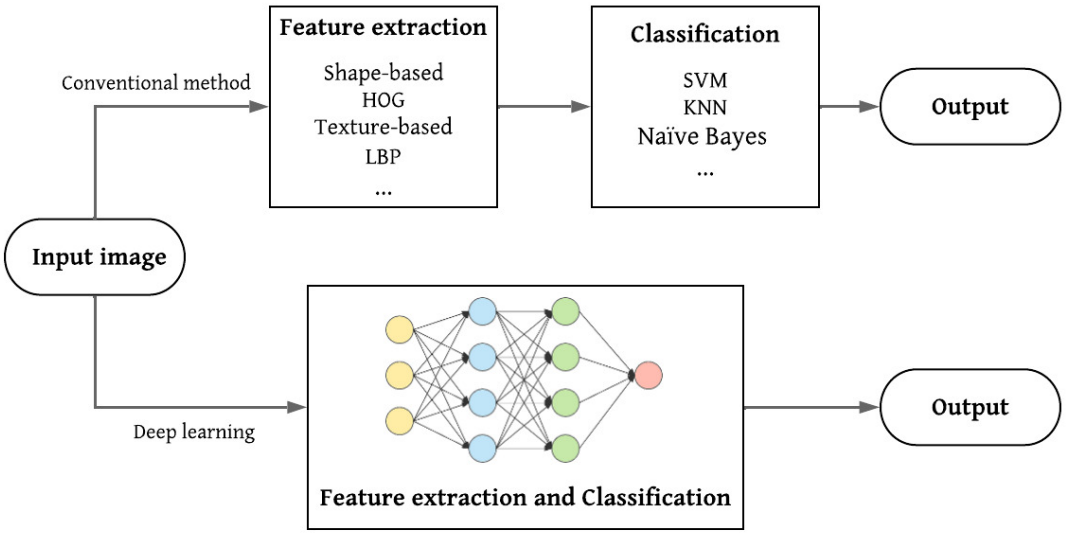


Figure 2.1: Workflow of image classification under two different approaches namely conventional and deep learning.

capable of learning hierarchical representations of the images, capturing both low-level features like edges and textures, as well as high-level semantic information.

2.1.1 Conventional image classification approaches

The performance of most conventional image classification methods relies on the combination of a feature extractor and a classification method. In essence, conventional methods operate by extracting features from input samples and then feeding these features into a classification algorithm to assign them to one of the target classes. There are various well-known feature extractors employed in conventional image classification. These include grapheme statistics, region segmentation features, edge continuity features, the Scale Invariant Feature Transform (SIFT)[84], Gray Level Co-occurrence Matrix (GLCM)[53], Local Binary Pattern (LBP)[97], Histogram of Oriented Gradients (HOG)[28], and more. A feature extractor transforms the raw image data into a mathematical representation called a feature descriptor. This descriptor is typically a vector

with one or more dimensions, known as a feature vector. The feature vector incorporates different informational components about an object. The level of detail required in the feature vector depends on what we want to learn or represent about the object. In the case of image classification, a feature vector represents the entire image. The feature vectors are then fed into a classifier, which is trained to perform the classification task. Some well-known conventional classification methods include Support Vector Machine (SVM), K-Nearest Neighbor (KNN)[25], Multi-Layer Perceptron (MLP)[57], Random Forest, Naïve Bayes[34], and others. These classifiers learn from the feature vectors and make predictions about the class labels of new, unseen images.

In summary, conventional image classification methods rely on the combination of feature extraction and classification techniques. Feature extractors transform the raw image data into feature vectors, which are then used to train classifiers to assign images to specific classes. The choice of feature extractor and classifier depends on the specific problem and dataset at hand.

2.1.2 Deep learning image classification approaches

Deep learning (DL) is a subset of machine learning (ML) and a branch of artificial intelligence (AI) that enables machines to learn from data. Figure 2.1 illustrates how deep learning utilizes neural networks for various tasks. In this context, neural networks are systems composed of interconnected nodes or neurons. These networks process input data through hidden layers, where each node performs computations and passes the results to the next layer. The process continues until reaching an output layer, which provides the final prediction. Different types of neural networks exist based on the operations performed within the hidden layers. One widely used type for image classification is the Convolutional Neural Network (CNN). This section focuses on explaining the working principles of CNNs.

CNNs are specialized neural networks designed to extract distinctive features from image data. Compared to other classification algorithms, CNNs require minimal preprocessing of the input data. They consist of three main types of layers: (i) convolutional layer, (ii) pooling layer, and (iii) fully-connected (FC) layer. The first layer in a CNN is the convolutional layer. It applies filters or kernels to the input image, performing convolutions that detect specific patterns or features. These convolutional operations capture local spatial relationships in the image, allowing the network to learn relevant visual features. Following the convolutional layer, there may be additional convolutional layers or pooling layers. Pooling layers reduce the dimensionality of the data by summarizing and downsampling the information captured by the previous layers. This reduces the computational complexity of the network and helps in capturing the most important features. The final layer of a CNN is the fully-connected layer. This layer takes the features extracted by the previous layers and applies them to a traditional neural network structure. It connects all the neurons of the previous layer to the output nodes, enabling the network to make predictions based on the learned features. As data passes through the layers of a CNN, the network becomes more capable of detecting larger areas or shapes within the image. Early layers focus on basic elements like colors and edges, while deeper layers recognize more complex parts or shapes of the object until the correct classification is achieved.

In summary, CNNs are specialized neural networks that excel in extracting features from image data. They consist of convolutional, pooling, and fully-connected layers, working together to recognize patterns and make accurate predictions. By leveraging the hierarchical nature of the network, CNNs can effectively learn and distinguish various visual features, enabling successful image classification tasks.

- *Convolutional layer*

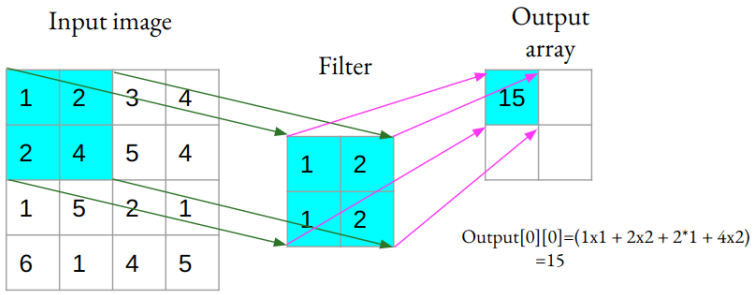


Figure 2.2: Working of convolution operation.

The convolutional layer is a central component of a CNN and performs the majority of computations. It requires input data, typically a color image composed of a 3D pixel matrix. The key element in this layer is the feature detector, also known as a kernel or filter. The feature detector moves through the image's receptive fields to determine the presence of specific features, a process known as convolution.

In the convolutional layer, a 2D array of weights acts as the feature detector and operates locally on the image. The size of the filter, which determines the receptive field size, is typically a 3x3 matrix but can vary. The filter is applied to a portion of the image, and the dot product between the input pixels and the filter is computed. This dot product is then fed into the output array. The filter continues to move through the receptive field with a certain stride, repeating the dot product operation. A feature map is produced by a sequence of dot products between the input and the filter. Figure 2.2 visualizes the first step of this process. It is important to note that in the feature map, the pixel values of the input image do not need to be fully connected. Only the receptive field, where the filter is applied, needs to be connected. Convolutional layers (as well as pooling layers) are often referred to as "partially connected" layers because each value in the output array is not directly linked to every value in the input array.

The idea of parameter sharing describes how, during the convolution process, the weights in the feature detector stay constant as it traverses over the image. Gradient descent and backpropagation techniques are used during training to modify some parameters, such as weight values. Three hyperparameters, nevertheless, must be set before the neural network is trained since they have an impact on the output's volume. These hyperparameters include:

- Filter count: Determines the size of the receptive field and the filter matrix. Common sizes are 3x3 or 5x5, but other dimensions are possible.
- Stride: Defines the step size by which the filter moves across the input image. A stride of 1 moves the filter pixel by pixel, while a stride of 2 skips one pixel in between.
- Padding: In some cases, padding is applied to the input image before convolution to preserve the spatial dimensions of the output feature map. The padding adds extra border pixels around the input image.

Setting these hyperparameters appropriately is crucial for achieving desired output volumes and controlling the spatial information flow through the network.

In a CNN, non-linear operations like ReLU[50] are applied to the feature map after each convolution to introduce nonlinearity. Multiple convolutional layers in a CNN create a hierarchical structure, allowing later layers to access information from earlier layers' receptive fields. This hierarchy enables the network to capture features at different levels of abstraction, leading to better performance in tasks like image classification. The combination of hierarchical structure and non-linear activations allows CNNs to model complex relationships and extract high-level representations from the input data.

- *Pooling layer*

Dimensionality reduction and a reduction in the number of input parameters are done via the pooling layer, also known as downsampling. The pooling operation applies a filter to the entire input, just like the convolutional layer. The pooling filter, in contrast to the convolutional layer, does not have weights. Instead, the pooling layer applies an aggregation function on the values in the receptive field to fill the output array. There are primarily two types of pooling:

- Max Pooling: Max pooling selects the maximum value from the receptive field and assigns it to the output array. This helps in capturing the most prominent features within the receptive field while discarding less important details. Max pooling provides translation invariance, meaning that the model can still recognize the features regardless of their precise location within the receptive field.
- Average Pooling: Average pooling calculates the average value of the receptive field and assigns it to the output array. This helps in smoothing out the features and reducing noise in the data. Average pooling can be useful in scenarios where preserving precise details is not necessary, such as in some image classification tasks.

The spatial dimensions of the feature map can be decreased with the aid of both max pooling and average pooling, which lowers computing cost and controls overfitting. In order to gradually shrink the spatial size of the representations while preserving the most important data, pooling layers are often included between succeeding convolutional layers in a CNN. Despite losing some information, the pooling layer of a CNN offers a number of benefits. It lessens complexity, boosts efficiency, and lessens the danger of overfitting.

- *Fully-Connected (FC) Layer* The FC (Fully-Connected) layer in a neural network is so named because every node in the output layer is directly connected to a node in the layer above it. Unlike the convolutional and pooling layers, which have partial connections, the FC layer establishes direct connections between all nodes. The FC layer performs the classification task based on the features extracted from the preceding layers and their respective filters. To assign inputs to the correct categories, FC layers often employ a softmax activation function, which produces a probability value between 0 and 1.

There exist several CNN models, including VGGs[125], ResNet (Residual Neural Network)[55], Inception[135], and more, which are available in Keras(some of these models are discussed in Appendix A.1). Keras provides pretrained variants of these models trained on the ImageNet dataset[98], which consists of billions of samples classified into 1000 classes. However, in many domains, data scarcity can be an issue for training machine learning models. To address this problem, deep learning introduces the concept of transfer learning, where pretrained models (such as pretrained VGG models provided by Keras) can be reused for similar tasks. The process of reusing pretrained models for transfer learning is discussed in Section 2.3.

2.2 Image Segmentation

The technique of separating an image into different sections of interest is known as image segmentation. This procedure, known as document layout analysis when applied to document images, seeks to distinguish between text and non-text components. The text and non-text components segregation, text region identification, text recognition and information extraction, and vanquishing of non-textual components are all interchangeable when referring to this activity. Over the years, numerous studies have proposed document layout analysis systems based on traditional machine learning (ML) methods,

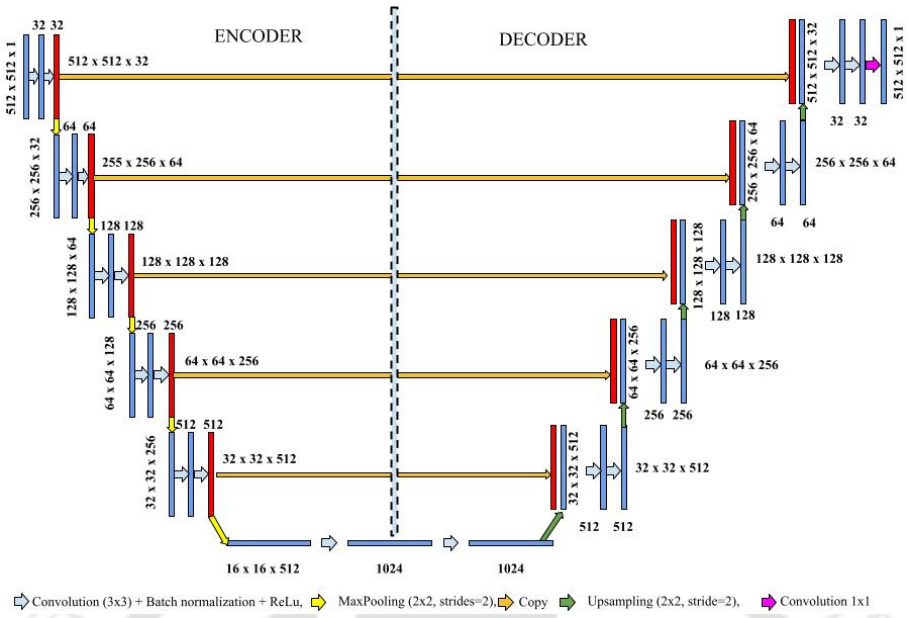


Figure 2.3: Architecture of image segmentation model, UNet (Proposed in Study[109]).

deep learning (DL) methods, and rule-based learning methods. Chapter 3 provides a brief survey of these approaches. The segmentation of documents into multiple regions of interest is a crucial step in performing document layout analysis. Image segmentation involves assigning a pixel-wise mask to each object present in the image. Unlike traditional image classification, where a single label is assigned to the entire image, image segmentation predicts the class for each pixel, and the predicted classes are assigned unique colors for visualization purposes. In this thesis, we have developed a segmentation framework (discussed in Chapter 3) based on a well-known CNN-based segmentation model called UNet[109]. This section will provide further details on the working principles of UNet.

UNet, originally introduced in 2015, was developed as a modification of conventional CNNs specifically for processing biomedical images[109]. While a standard CNN focuses on classifying images by assigning a single label to the entire image, biomedical applications often require identifying both the presence of a disease and the precise location

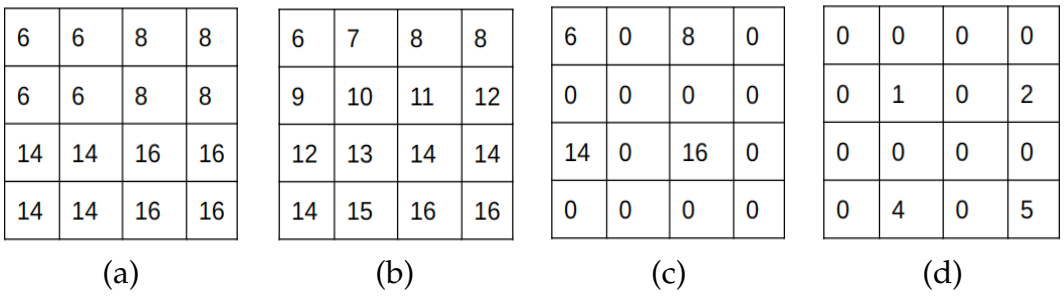


Figure 2.4: Upsampled feature map of different Unpooling techniques: (a) Nearest Neighbors, (b) Bilinear interpolation, (c) Bed of Nails, (d) Max Unpooling.

of abnormalities within the image. UNet addresses this challenge by performing pixel-level classification, enabling the localization and distinction of borders within the image. In UNet, the input and output share the same size, and classification is performed on every pixel. The architecture of UNet is depicted in Figure 2.3 and follows a symmetric structure with two main components: the *Encoder* and the *Decoder*. In the Encoder part, the image is downsampled, capturing and extracting high-level features. These features are then passed to the Decoder part, where the image is upsampled, and the spatial information is gradually regained. This symmetric structure helps in capturing both local and global information, allowing for accurate segmentation and localization of objects in the image. The downsampling is done by pooling or strided convolution, but for the upsampling stage in UNet, two common methods are typically employed:

- *Unpooling:* Unpooling is the counterpart of pooling, aiming to restore the original input feature map. It operates by taking the largest or average response from each subdivided region of the feature map. Let's consider a feature map of size 4×4, $A = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]$, and it downsampled feature map as $A_d = [[6, 8], [14, 16]]$. The explanation of several techniques used for upsampling the downsampled feature map, A_d are discussed below:
  - Nearest Neighbors: This method takes an input pixel value and copies it to

the K-nearest neighbors, where K depends on the desired output. However, this approach can result in blocky output structures as all pixels within each sub-region have the same value. The output upsampled feature map can visualized in Figure 2.4 (a).

- Bilinear Interpolation: Bilinear interpolation employs linear interpolation in both directions, averaging the weights of the four neighboring pixels and assigning them to the empty pixels. This helps in achieving a smoother output structure. The output upsampled feature map of A_d can visualized in Figure 2.4 (b).
- Bed of Nails: In this technique, the input feature response is placed in the top-left corner of the corresponding sub-region in the unpooled output, while setting all other elements in the sub-region to zero. This approach generates a fine-grained output structure, but the upsampled elements always have a fixed location, typically the upper-left corner of each sub-region. The output upsampled feature map of A_d can be visualized in Figure 2.4 (c).
- Max Unpooling: Max Unpooling addresses the limitations of the Bed of Nails approach. It performs upsampling similar to Bed of Nails but also remembers the indices of the sources of the largest elements before max pooling. During the Max Unpooling process, this information is utilized to reposition the elements in each sub-region to their original locations prior to max pooling. The output upsampled feature map of A_d can be visualized in Figure 2.4 (d).

- *Transpose Convolution:* Transpose convolution, also known as deconvolution, is a learning-based approach to upsampling. It can be considered as the reverse process of a simple convolutional operation. Transpose convolution involves learning

the weights during training, allowing the network to adaptively upsample the feature map based on the given task.

Both unpooling and transpose convolution methods contribute to restoring the spatial information lost during the downsampling stage, enabling the decoder part of UNet to reconstruct the original input size while preserving the necessary details for accurate segmentation.

In the UNet architecture which is shown in Figure 2.3, the input image size is $512 \times 512 \times 1$, and it undergoes downsampling in the encoder part.The encoder portion increases the number of channels or feature maps while decreasing the spatial dimensions of the image. The input image is downscaled in order to extract abstract representations and capture high-level features. The expanding path or decoder portion of the network, which is located at the bottom, is responsible for upsampling the image to its original size. It attempts to generate high-resolution output from the encoder's low-resolution feature maps. During the upsampling process, the image is initially upsampled from 16×16×1024 to 32×32×512. This upsampling is performed using transposed convolutional layers or other upsampling techniques. The purpose is to restore the spatial dimensions of the image while reducing the number of channels. After the initial upsampling, the upsampled image is concatenated with the corresponding image from the encoder part. This concatenation allows the network to combine the low-level and high-level features, enabling more precise predictions. The resulting concatenated image has a size of $32 \times 32 \times 1024$. By combining the information from both the encoder and decoder parts, the UNet architecture can effectively capture both local and global information, making it suitable for tasks such as image segmentation.

Like any other supervised method, UNet takes two inputs viz. input images and their corresponding ground truth which is the masked images. Figure 2.5 shows the sample of some input images and their respective masked images where the regions of

Figure 2.5: Input samples for the segmentation model, UNet where only the textual regions are the regions of interest.

interest are only textual components leaving non-textual components. The masking of input images can be achieved with various tools such as LabelBox∗ and keylabs†.

2.3 Transfer learning for deep neural network

Transfer learning is a potent deep learning technique that uses an existing model that has been trained on one task with lots of labeled data as the foundation for another task with little to no training data. Transfer learning enables us to use the information and patterns discovered during the previous assignment to enhance performance on the

∗https://labelbox.com/
†https://keylabs.ai/image-annotation-tool.php

current task rather than creating a model from the start. Time and computational resources are saved by using this strategy. Transfer learning comes in two different varieties depending on how knowledge is transferred:

- Feature Extraction: The pre-trained model is employed as a fixed feature extractor. Only the last layers, also known as the classifier or fully connected layers, of the pre-trained model are updated or retrained for the current job, with the earlier layers being kept frozen. The pre-trained model has already acquired useful features from a sizable dataset, and the new task can make use of these features. We may take advantage of the pre-trained model's capacity to extract pertinent and high-level features by applying feature extraction while customizing the final layers to the particular task at hand.
- Fine-tuning: Not only are the final layers of the pre-trained model updated or retrained during fine-tuning, but some of the earlier layers are also unfrozen and permitted to be fine-tuned or retrained on the new job. We improve the model's ability to adapt to the new task and potentially learn task-specific properties by adjusting the earlier layers. More training data are needed for fine-tuning, which should be done cautiously to prevent overfitting, as the model may easily forget the previously learned information if the new job is significantly different.

In transfer learning, the choice between feature extraction and fine-tuning depends on a number of variables, such as how similar the new job is to the task that was previously taught, the availability of training data, and the available computational resources. When the new task has little data or differs greatly from the pre-trained task, feature extraction is frequently used. With this method, just the final layers relevant to the new task are added and trained, and the pre-trained model is used as a fixed feature extractor. Feature extraction enables the extraction of significant characteristics that

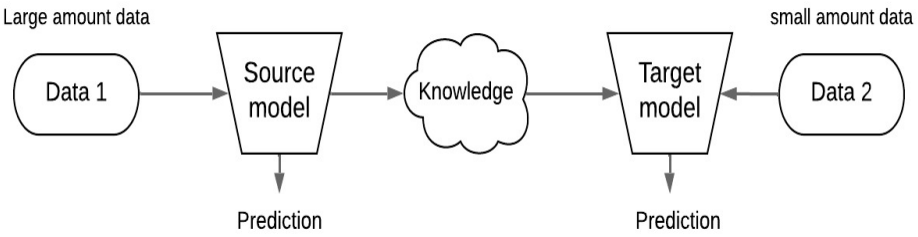


Figure 2.6: Workflow of transfer learning.

may be applied to the new job by utilizing the learned representations from the pre-trained model. On the other hand, fine-tuning works better when there is enough training data and the new task is similar to the one that was previously taught. The pre-trained model's early layers are additionally trained on the new task during fine-tuning, in addition to its final layers. As a result, the model can modify and improve the learnt representations to better fit the requirements of the novel task.

The choice between feature extraction and fine-tuning should be made based on the specific requirements and constraints of the new task. If the new task has limited data or is substantially different from the pre-trained task, feature extraction provides a good starting point. However, if the new task is similar to the pre-trained task and sufficient data is available, fine-tuning can potentially lead to further improvements in performance. Transfer learning, in general, is a valuable technique in deep learning as it enables us to leverage the knowledge and patterns learned from one task to improve the performance of a new task, thereby accelerating the learning process and achieving better results.

2.4 Attention on Convolutional Neural Network

To enhance the performance of convolutional neural networks (CNNs), researchers have explored various factors such as depth, width, cardinality, and attention. Attention is

an advanced technique that can significantly improve the representation capability of CNNs by capturing long-range feature interactions. There are two main ways in which attention can be applied in computer vision:

- Spatial Attention: Spatial attention focuses on determining where to focus within an image. It assigns weights to different spatial locations, highlighting the most relevant regions or pixels. This allows the network to selectively attend to important spatial features while suppressing less informative regions. Spatial attention mechanisms can be used to enhance the discriminative power of the network and improve its ability to localize and recognize objects within an image.

- Channel Attention: Channel attention operates at the level of feature channels in the network. It aims to dynamically adjust the importance of different channels to better capture relevant information. By learning channel-wise attention weights, the network can emphasize the most discriminative and informative channels while suppressing less useful ones. Channel attention helps in enhancing the representation capability of the network and improving its ability to capture important features and patterns across different channels.

By incorporating attention mechanisms, CNNs can effectively capture long-range dependencies and focus on relevant regions or channels, leading to improved performance in various computer vision tasks such as image classification, object detection, and image segmentation. The use of attention mechanisms provides the network with the ability to allocate its resources more effectively and adaptively, resulting in enhanced feature representation and overall performance.

In the realm of computer vision, there are two widely recognized attention modules: the Squeeze and Excitation Network (SE)[59], and the Convolutional Block Attention Module (CBAM)[149]. These attention modules have been extensively studied and proven

effective in various computer vision tasks. In the following sections, we will delve into a detailed discussion of these two attention modules, exploring their mechanisms and capabilities.

2.4.1 Squeeze and Excitation Network (SE)

The Squeeze and Excitation Network (SE) attention mechanism is introduced by study[59]. It focuses on providing attention to the channel. It consists of two primary operations: squeeze and excitation. These operations are described below:

- Squeeze operation: In this step, the spatial information is reduced to capture global context by performing global average pooling. It reduces the spatial dimensions of the feature maps while retaining channel-wise information. Mathematically, for an input feature map, denoted as F, the $R = GlobalAvgPool(F)$. Here, R represents the channel-wise information obtained by globally averaging the values of each channel.

- Excitation operation: This operation models the channel-wise dependencies and captures the importance of each channel by learning channel-wise attention weights. It applies a gating mechanism to the channel-wise information obtained from the squeeze operation. Mathematically, the excitation operation can be represented as $T = \gamma(W_2\eta(W_1, R))$. Here, W_1 and W_2 are learnable parameters (weights) of fully connected layers, η is the activation function (in general, the ReLU is considered), and γ presents the sigmoid function. T represents the channel-wise attention weights, indicating the importance of each channel.

- Scale and reweighting: Finally, the original feature map is multiplied element-wise with the channel-wise attention weights obtained from the excitation operation to emphasize important features. Mathematically, the output feature map Y can

be obtained as $Y = F \odot T$, where $\odot$ represents the element-wise multiplication.

The SE module can be inserted into a CNN architecture by integrating it into the network's convolutional layers. It enables the network to adaptively emphasize informative channels and suppress less relevant ones, leading to an improved representation of learning and performance.

2.4.2 Convolutional Block Attention Module (CBAM)

The Convolutional Block Attention Module (CBAM) attention mechanism is introduced in study[149]. It offers attention to both spatial and channel dimensions. The mechanism consists of two modules that are sequentially connected: the channel attention module and the spatial attention module.

- Spatial Attention Module: The spatial attention module captures the spatial dependencies within feature maps. It helps the network focus on relevant spatial locations and suppress irrelevant regions. It takes an input feature map, denoted as F, and performs the following operations:
  - Max Pooling: The feature map F is fed into a max pooling operation, which downsamples the spatial dimensions to capture the most prominent features. Let's denote the output of max pooling as MP. This operation can be denoted as $MP = MaxPool(F)$.
  - Sigmoid filtering: The max-pooled feature map MP is passed through a convolutional layer followed by a sigmoid activation function. This generates a spatial attention map, denoted as S, with values between 0 and 1. The sigmoid function amplifies relevant spatial locations by assigning higher values and suppresses irrelevant regions by assigning lower values. This operation can be denoted as $S = Sigmoid(Convolution(MP))$.

- Channel Attention Module: The channel attention module captures interdependencies among different channels of the feature map. It allows the network to focus on important channels and suppress less informative ones. The channel attention module consists of two components: the global average pooling operation and the multi-layer perceptron (MLP). The channel attention module takes the original input feature map F and performs the following operations:

  - Global Average Pooling: The feature map F is globally average pooled, resulting in a feature descriptor vector. Let's denote this vector as V. It can be expressed as $V = GlobalAvgPool(F)$
  - Multi-layer perceptron (MLP): The feature descriptor vector V is passed through an MLP consisting of fully connected layers with non-linear activations. The MLP captures channel-wise relationships and generates channel-wise attention weights. Let's denote the output of the MLP as W. It can be expressed as $W = MLP(V)$

  Once the spatial attention map S and the channel attention weights W are obtained, they are combined(element-wise multiplication) to generate the final attention map A, i.e., $A = S \odot W$. Finally, the attention map A is element-wise multiplied by the original input feature map F, resulting in the attended feature map AF, i.e., $AF = F \odot A$.

In summary, CBAM combines spatial attention and channel attention modules to selectively enhance informative spatial locations and channel-wise features in the input feature map. This allows the network to focus on important regions and channels, leading to improved representation and performance in various computer vision tasks.

2.5 Optical Character Recognition (OCR)

An OCR system extracts text data from scanned documents, camera photos, and image-only PDFs. It extracts letters from the image, turns them into words, and then turns the words into sentences, allowing access to and alteration of the original texts. Additionally, it does away with the requirement for human data entry. Researchers across the globe have been developing OCRs for multiple languages and the research on this domain has more than 50 years of history. This study exploited the open source OCR tool "Tesseract OCR"[128] for the development of Manipuri Bengali OCR system. Therefore, in this section we are presenting the brief detail of Tesseract OCR and its ability for the development of new OCR system from the existing OCR of similar scripts.

Tesseract is available for a number of operating systems. It is a free software. Hewlett-Packard created the program as proprietary in the 1980s, and it was made available as open source in 2005 by Google. Since the development of its 4^{th} version in 2019, Tesseract OCR uses Long short term memory (LSTM). To date, it provides OCR models for 100+ languages and 35+ scripts. Tesseract provides multiple options for retraining the OCR model:

1. *Fine tune:* It retrains the existing OCR model on the specific additional data. This might be a effective for issues that are similar to the training data already available but differ in a subtle way, like a highly unique font. This approach may be effective even with little training data.

2. *Cutting off the top layer:* It retrains a new top layer with the fresh data by cutting out the top layer (or an arbitrary number of layers) from the network. This is most likely the best alternative after fine tune. If the process starts with the most similar-looking script, cutting off the top layer might still be effective for training the new script.

3. *Retrain from scratch:* This approach is suitable when there is sufficiently large training samples. If not, it will provide a network that is over-fitted and performs well on training data but poorly on real-world data.

This study exploits fine tune option to develop Manipuri OCR from the Bengali OCR. The process of fine-tuning the model is described in Appendix B.2.

2.6 Summary

This chapter presented the background studies on the research domains exploited by the works presented in this thesis. It includes image classification methods, image segmentation methods, transfer learning, attention modules, and a Tesseract-based OCR platform. With this background, the next chapter addresses the problems of document image segmentation and classification.

"Never doubt, when you begin with something that it will end in failure. Our thought is transformed as picture in our mind."

Napz Cherub Pellazo

3

Integrated Document Segmentation and Region Identification- Textual, Equation and Graphical

For any given document image, the first task of DCS (as discussed in Chapter 1) can be categorized into two sub-tasks: (i) document segmentation: segmentation of the regions of interest (textual or non-textual), and (ii) meta-file generation: save the size

of the given document image, positional information of the segmented true regions of interest, and further save the segmented true regions of interest as individual images. This chapter presents the development of the segmentation model to perform the first sub-task. As the second sub-task, which is the generation of a meta-file and its updating process, is a continuous process till the end of the pipe-line of DCS, it is presented after discussing all the sub-modules of DCS.

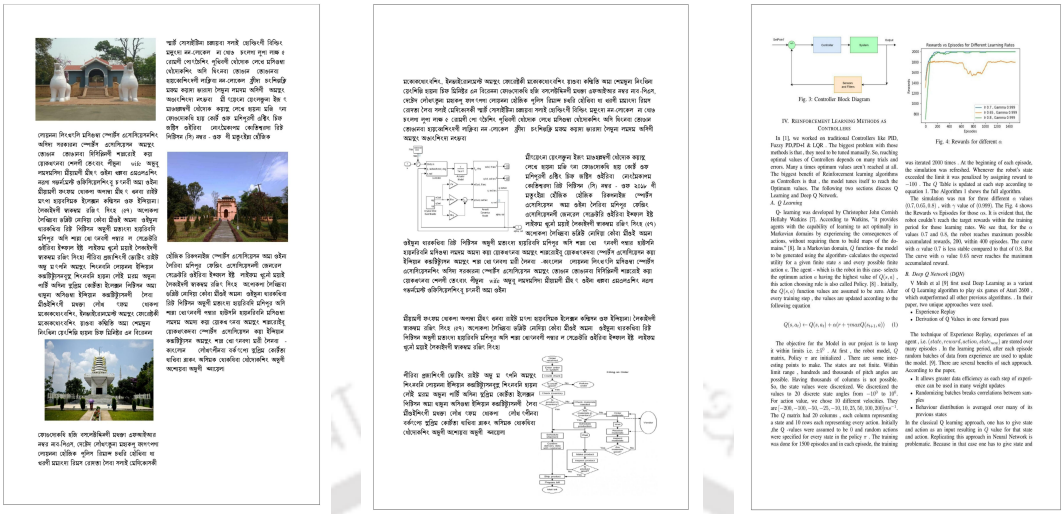
3.1 Introduction

With the increase in the availability of digital image documents publicly on digital platforms, digital image document analysis is becoming an important research problem for applications like information retrieval, document classification, and OCR. Document segmentation and region classification[113,37] are some of the core issues for a document image analysis task. While digital image document analysis needs segmentation of both textual and non-textual regions, the majority of the studies focus on identifying individual regions of interest. For instance, the studies[90,104,80,144] focus on identifying only textual regions. Whereas, the studies[113,100] focus only on identifying non-textual regions. On the other hand, a few studies[142,144,151] have attempted to identify both the textual and non-textual regions jointly with a single model. From the brief review and empirical studies, the following two points are noted: (i) most of the integrated models adopt a single model to segment different class regions (textual or non-textual) of interest; and (ii) most of the integrated models fail to identify non-textual regions effectively when the majority of the pixels in the regions are sparsely clustered. A pixel can be said to be sparse if it is distributed in such a way that its neighboring pixel can be found through only 4 neighbors' connectivity. Figure 3.1 shows the comparison of documents that have non-textual regions with sparsely clustered pixels and tightly clustered pixels (a pixel can be said to be tightly clustered if it is distributed in such a

way that its neighboring pixel can be found through 4 neighbors' connectivity or diagonal connectivity or 8 neighbors' connectivity). The performances of three existing methods over the document samples that have non-textual regions with sparsely clustered pixels are shown in Figure 3.2. The figure shows that all of the methods correctly segment and identify textual regions but fail to correctly segment non-textual regions. It may also be noted that many of the non-textual regions in scientific documents are made up of sparsely clustered pixels.

Motivated by the above observations, this study proposes a 2-tier feedback-based end-to-end integrated framework which can jointly segments and identifies regions of interest accurately. The regions of interest are classified into three major classes: *textual, equation*, and *graphical*. The regions that are considered under the class"graphical" may vary among studies, but this study considers all the regions other than equations, and textual as "graphical" (such as natural scene images, data visualizations, flowcharts, and medical images). The first tier focuses on segmenting and identifying textual and equational regions, while the second tier segments and identifies the graphical regions∗. The use of two tiers primarily helps the model in providing better segmentation performance as it employs different segmentation models for different regions of interest. However, to address the issues of inaccurate segmentation of non-textual(graphical in our study) regions with sparsely clustered pixels, this study makes the two tiers asynchronous. The asynchronous is developed in such a way that the second tier depends upon the first tier : *First tier provides regional information (textual and equation) to the second tier so that the input document images are masked to have the majority of the foreground pixels as the graphical regions.* This approach of having two asynchronous and feedback-based tiers helps the segmentation models learn various characteristics of different regions of interest independently.

∗As textual and equations have similar characteristics, this study considers them as a single class while performing the segmentation.



(a) (b) (c)

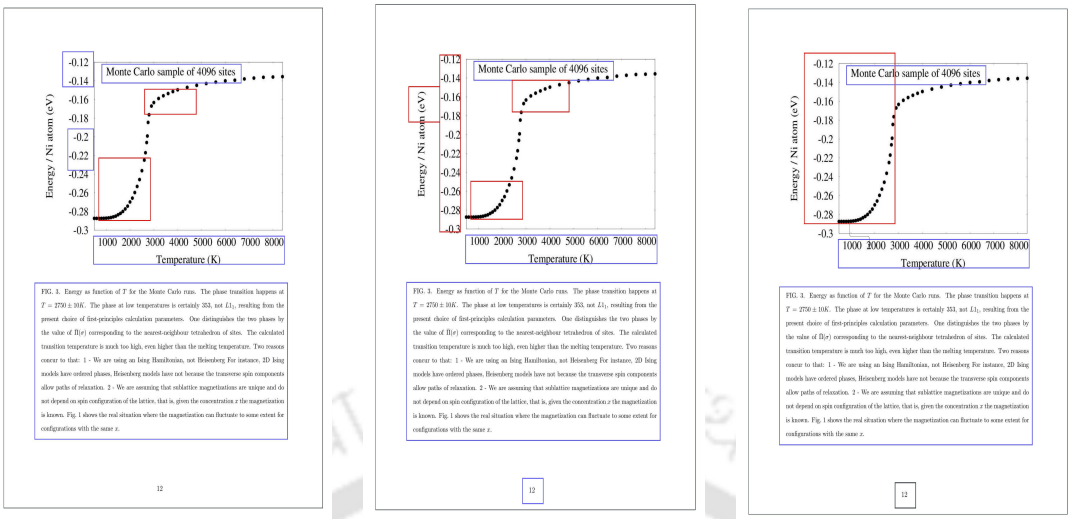
Figure 3.1: Samples of document images : The document sample in (a) has non-textual components in which the pixels are tightly clustered, while the samples in (b) and (c) has the non- textual components in which the pixels are sparsely clustered.

From the various experiments over five datasets (four publicly available and one locally generated), it is observed that the proposed framework outperforms various existing methods. The contributions of the thesis in this chapter may be noted as follows:

- Integrated document segmentation model: A 2-tier feedback-based end-to-end integrated model is proposed herein, which can effectively segment and identify non-textual components with sparsely clustered pixels.

- Dataset: The curated dataset (of 4840 samples), which consists of document images and their corresponding annotated masks, is available on proper request.

- The proposed framework was evaluated by considering four variants of the segmentation module, UNet, which is a widely utilized model for medical image segmentation.



(a) (b) (c)

Figure 3.2: Output Samples of three document segmentation methods: The samples in (a) (b), and (c) are the output of the method in study Tran, H.T et al.[142] , Umer, S et al.[144] and Wu, X. et al.[151].

- The evaluation of the proposed framework encompassed five distinct datasets, comprising four publicly available datasets and one locally generated dataset specifically for this study.
- Comparative evaluation of the proposed framework with three recently proposed existing integrated methods.

The rest of the chapter is organized as follows: A brief survey of the related studies is presented in Section 3.2. Section 3.3 presents the proposed framework. Sections 3.4 and 3.5 present the experimental setups and experimental evaluation of the proposed framework. Section 3.6 discusses the challenges of the proposed framework. Section 3.7 concludes the study presented in this chapter with a suggestion for future works.Finally, Section 3.8 summarizes this chapter's contribution to the development of the proposed DCS.

3.2 Related Studies

In the early stage of the document analysis system, text/non-text separation plays a vital role. Since the early 1980s, a significant number of published articles have appeared dealing with the challenges of developing effective methods that are robust to varying document styles[80,40,14,144]. Based on the classification methods used by the existing studies, the journey of the document analysis system can be divided into two phases, viz. pre-deep neural network phase (1980s-2013) and the deep neural network phase (since 2014).

During the pre-deep neural network phase, most of the studies used two methods for text/non-text separation: *region classification-based* and *connected component (CC) classification-based*. A region classification-based method decides whether a segmented region is text or not, while a CC classification-based method does the same thing with the CCs. Region segmentation is generally performed using a top-down[40,14,6] bottom-up[131] or hybrid[21,129] approach. For top-down approaches, segmentation goes from a coarse level to a finer level, where large homogeneous regions are first extracted, and then refinement is performed at the subsequent levels. In bottom-up approaches, the processing is performed in the reverse direction, i.e., from the level of pixels to the regions. In the bottom-up approach, the process begins with local information. It starts with merging pixels to form characters, characters to words, words to text lines, and then text lines to regions. In connected component (CC) classification-based methods, CCs are extracted from the input document image, and then each CC is classified as text or non-text. In this era, to capture structural information, all approaches rely on hand-crafted features such entropy[99,83,131], homogeneity[99,143], energy[99,131], Mathematical morphology[99], and white tile-based texture[3,121].

During the deep learning era, most of the existing studies exploited various CNN-based models. Using CNN-based models, various studies focus not only on the segmen-

tation of the document but also the classification of the document as a whole[70], word clustering[78]. Study[70] learns hierarchical features directly from normalized image pixels. It employs ReLU and dropout to enhance the training of CNN. It classifies document images into ten classes such as ads, news, reports, email, and application forms. With a theory that the low-dimensional features contain more detailed information and the high-dimensional features contain more semantic information, the study[151] proposed an end-to-end united network named Dynamic Residual Fusion Network. Study[144] proposed a deep CNN model which divides the input document to multiple sizes patches to handle various size of fonts. It basically follows three phases, preprocessing, classification of text/non-text, and post-processing. Some studies proposed deep learning approaches that focus only on the detection of some specific graphical components, such as tables[123,49], and formulas[45].In most of the studies, document segmentation is carried out by a single encoder-decoder segmentation model. The study[142] proposed a model for text segmentation and non-text classification in document images through a deep learning UNet segmentation model. It employs one segmentation model each for textual and non-textual components, which performs segmentation in a synchronous manner in a parallel fashion.

The 2-tier feedback-based framework proposed in this thesis is a deep learning-based approach. Both tiers consist of document segmentation and classification using CNN-based models, followed by post-processing methods. The proposed framework is different from most of the existing methods in two aspects: (i) employment of two asynchronous tiers to segment and classify different regions of interest (textual and equation in the first tier and graphical region in the second tier) (ii) masking of the input document images (in the second tier) using the feedback information provided from the first tier. With these two aspects, the proposed framework addresses the issue of inaccurate segmentation of graphical regions reported in various existing studies. The

proposed framework could be considered similar to the study in[142] because of the two tiers, post-processing, and CNN-based segmentation models. However, as mentioned earlier, our proposed approach adopts an asynchronous feedback flow from the first to the second tier, which helps the second tier to act according to the received information. In addition to this, the study in[142] dedicated its post-processing algorithm only to contour-based information, which sometimes fails to filter out false segmented regions of interest, which is addressed in our proposed method considering the CNN-based classification model on the top of contour-based information.

3.3 Proposed Framework

Considering the need to segment the regions of two contrasting natures, namely *textual (or equation)* and *non-textual (graphical)*, which may compose of regions with varying degrees of pixel distributions, our proposed method uses a two-tier approach to handle them separately. The first tier focuses on segmenting textual and equation regions, whereas the second tier focuses on segmenting graphical regions. These two tiers are connected by the regional information (co-ordinates of the textual and equation regions). The information flows from the first tier to the second tier to properly identify graphical regions of interest that may have sparsely clustered pixel distributions. The proposed model is explained in detail in this section, divided into two parts; *(i) textual region segmentation and identification* and *(ii) graphical region segmentation and identification*.

3.3.1 Textual region segmentation and identification

Given an image, the goal of this module (the first tier) is to segment and identify both textual and mathematical equation regions. Figure 3.3 shows the schematic diagram of this module, consisting of three sub-steps: *(i) region segmentation (ii) region classification* and *(iii) pruning of false textual regions*.

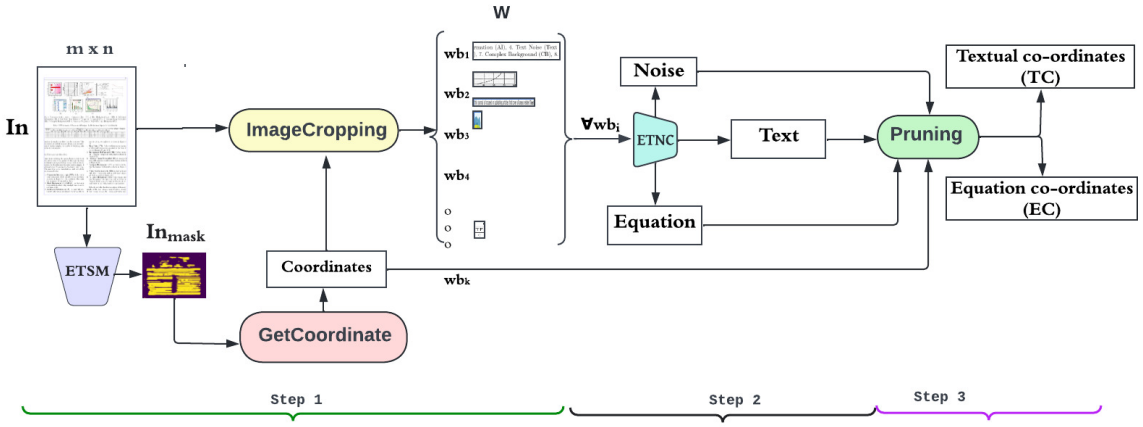


Figure 3.3: Equation and textual region segmentation and identification.

- *Region segmentation :* Given an image, the first task is to develop a textual region segmentation model (we called it as *Equation and Textual Segmentation Model (ETSM)*). We consider UNet[109], a well-known image segmentation model (with VGG-16[125] as a backbone in the encoder), as the model to segment textual and equation regions. The ground truth masked images are defined by considering only textual and equation regions as the regions of interest. It can be illustrated as below.

$$Y' \leftarrow ETSM(X, Y) \tag{3.1}$$

where Y' is the output masked images and X is the set of training document images, and Y is the set of corresponding masked images. (More detail about the dataset is provided in Section 3.4.1). As the learning parameters, cross-entropy loss function and Adam optimization method are used. The trained ETSM produces a masked image, In_{mask}, for each of the given document images. To get the position of the predicted regions of interest in In_{mask}, this study finds the co-ordinates of the bounding boxes of the predicted regions. It is done in the function **GetCoordinate** (as shown in Figure 3.3). It consists of three operations:

– *Contour detection in the mask:* Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. As a result, the number of contours in In_{mask} equals the number of segmented regions. Contours can be determined as follows

$$C = getContour(In_{mask}) \tag{3.2}$$

where *getContour*() is the algorithm described in Study[133]* for obtaining contours, and C is the list of contours $C = [c_1, c_2, c_3..c_k]$ presented in In_{mask}. The contour c_i consists all the boundary points of the i^{th} segmented regions, and denoted as $c_i = (\mathbf{x}, \mathbf{y}) = \{(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), ..., (x_{if}, y_{if})\}$.

– *Contour to coordinate conversion:* The co-ordinates of a bounding box are denoted by a tuple (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) are the top left and right bottom co-ordinates of the box. As done in the study[114], this study extracts the co-ordinates (x_i1, y_i1, x_i2, y_i2) of bounding box (b_i) for the region r_i from its contour c_i. It can be visualized as shown below.

$$(x_i1, y_i1, x_i2, y_i2) = min(\mathbf{x_i})-0.5, min(\mathbf{y_i})-0.5, max(\mathbf{x_i})+0.5, max(\mathbf{y_i})+0.5 \tag{3.3}$$

The addition of 0.5 in the equation ensures that the segmented region obtained from the bounding rectangle includes the complete desired region and has the proper spatial extent. By adding 0.5, the rectangle boundaries are shifted slightly to align more accurately with the underlying pixel grid, resulting in a more accurate representation of the desired segmented region.

With equation 3.3, the co-ordinates of all the bounding boxes, b_i for all re-

*The contour detection method exploited in the study[133] depends on the counting of 1-components and extraction of borders from a binary image.

gions r_i are extracted from their respective contours c_i and stored in a list $CD = [cd_1, cd_2, ...cd_k]$, where $cd_i = (x_{i1}, y_{i1}, x_{i2}, y_{i2})$.

- *Scaling up the co-ordinates:* The bounding box co-ordinates in the list CD are calculated with respect to the mask image In_{mask}. However, we want the bounding boxes in the input document images (which can be of arbitrary size $m \times n$). Let $a \times b$ be the size of generated masks image In_{mask}, then the scaling up of the coordinate for any $cd_i \in CD$ is done as follows

$$(x_1, y_1, x_2, y_2) = (x_1 \times Rx, y_1 \times Ry, x_2 \times Rx, y_2 \times Ry) \tag{3.4}$$

where Rx, and Ry are the scaling factor which are defined as $\lceil n/b \rceil$, and $\lceil m/a \rceil$, respectively.

Once the co-ordinates of the segments are estimated, the model further crops and extracts the corresponding segmented images $W = (wb_1.jpg, wb_2.jpg, ...wb_k.jpg)$ from the input image for further classification and pruning. This process is denoted by **ImageCropping** in Figure 3.3. Its operation is to slice the input image with the given co-ordinates. For any given coordinate $co-ordinates_i$, wb_i is obtained with

$$wb_i = In[(x_{i1} : x_{i2}), (y_{i1} : y_{i2})] \tag{3.5}$$

where $(x_{i1} : x_{i2})$ indicates the start and end points with respect to the row of the images, while the second parameter $(y_{i1} : y_{i2})$ indicates the start and end points with respect to the column of the images.

- *Region classification:* Once images of the segments are extracted from the input image, in the previous step, the next task is to classify the type of each segmented image. A CNN based classifier, as defined in Table 3.1 is used for identifying the

Table 3.1: Architecture detail of two classifiers: ETNC and GC (GC to be discussed in Section 3.3.2).

Layers	Output shape	Parameters #
Input layer	64 × 64× 3	0
Convolution 2D	64 × 64 × 32	896
MaxPooling 2D	32 × 32 × 32	0
BatchNormalization	32 × 32 × 32	128
Dropout	32 × 32 × 32	0
Convolution 2D	32 × 32 × 32	9248
MaxPooling 2D	16 × 16 × 32	0
BatchNormalization	16 × 16 × 32	128
Dropout	16 × 16 × 32	0
Flatten	8192	0
Dense	512	4194816
BatchNormalization	512	2048
Dropout	512	0
Dense	256	131328
BatchNormalization	256	1024
Dropout	256	0
Dense	3	514
Total Parameters	**4340130**	

type of image segments, which we name as *Equation Textual and Noise Classifier* (ETNC) in Figure 3.3. The CNN model is trained with cross entropy loss function and Adam optimizer. Given an image segment wb_i, the class assignment is illustrated as follows.

$$class_i = \begin{cases} \text{Equation}, & \text{if } ETNC(wb_i) = Equation \\ \text{Text}\ , & \text{if } ETNC(wb_i) = Textual \\ \text{Noise}, & Otherwise \end{cases} \tag{3.6}$$

where Equation denotes a segment with equation components, Textual denotes a segment with textual components, and Noise denotes segment other than textual and equation.

- *Pruning of false textual regions:* Once the image segments are classified into three classes in the previous step, the next step is to filter out the segments that are classified as Noise, and prune the false regions of interest from the segments

Algorithm 1 Pruning(Class, co-ordinates)

Input: co-ordinates, and Class
Output: Two sets of co-ordinates: Textual co-ordinates, and Equation co-ordinates
for each co-ordinates$_i$ in co-ordinates **do**
  if $class_i$ ==*Noise* **then**
    Remove co-ordinates$_i$ and update the set co-ordinates
  end if
end for
for each co-ordinates$_i$ ∈ co-ordinates **do**
  $Width = x_{i,2} - x_{i,1}$
  $Height = y_{i,2} - y_{i,1}$
  $A = Width \times Height$
  $Ar = Width/Height$
  $Area$.insert(A)
  $Aspect - ratio$.insert(Ar)
end for
Normalised_Area=$\{\acute{a}_1, \acute{a}_2, ..\}$ where $\acute{a}_i = \frac{a_i \in Area}{\sum_{a_j \in Area}}$
Normalised_Aspect-ratio=$\{\acute{ar}_1, \acute{ar}_2, ..\}$ where $\acute{ar}_i = \frac{ar_i \in Aspect-ratio}{\sum_{ar_j \in Aspect-ratio}}$
for each co-ordinates$_i$ ∈ co-ordinates$_i$ **do**
  if $a_i > theta$ **then**
    if $class_i$ ==*Text* **then**
      Add co-ordinates$_i$ in Textual co-ordinates **else** Add in Equation co-ordinates
    end if
  else
    if $ar_i < delta$ **then**
      if $class_i$ ==*Text* **then**
        Add co-ordinates$_i$ in Textual co-ordinates **else** Add in Equation co-ordinates
      end if
    end if
  end if
end for
Return [Textual co-ordinates, Equation co-ordinates]

that are classified as Text, or Equation. Most of the false regions of interest are embedded textual regions, such as text inside figures (representing labels), inside

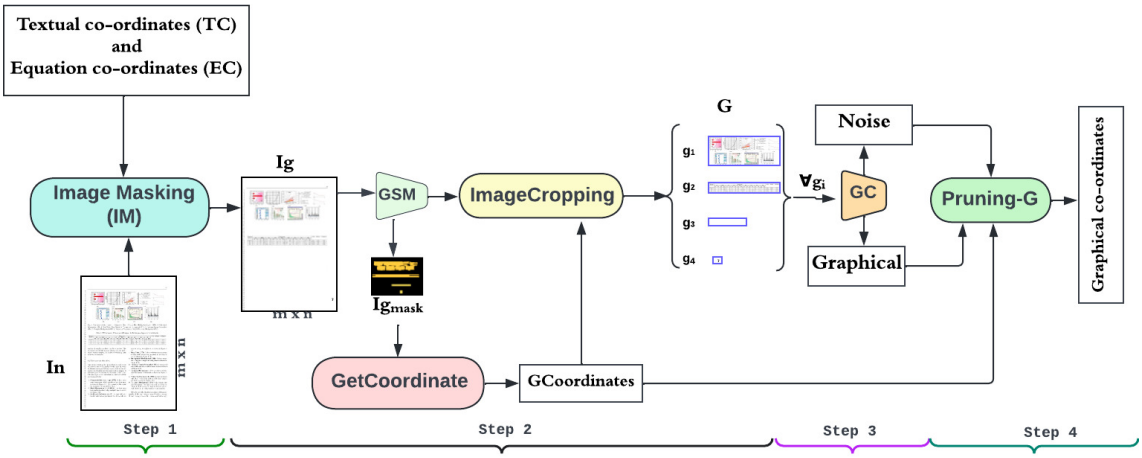


Figure 3.4: Graphical region segmentation and identification.

tables (representing labels, and entries), line numbers, and page number. The pruning is done considering two parameters of the regions, which are (i) Aspect ratio: most of the true regions (textual and equation) of interest are elongated horizontally as compared to the false regions of interest. So, the aspect-ratio (width/height of the region) is larger for true regions of interest. (ii) Area: area of the region for false regions of interest are always small as compared to the true regions (textual and equation)of interest. As shown in Figure 3.3, the process of this step is carried out in **Pruning**, which is described in Algorithm 1. It takes two inputs; (i) Class vector - the class labels of each image segment produced by ETNC i.e., $\textbf{class}_i$ denotes the class label of wb_i image segment, and (ii) co-ordinates vector - co-ordinates of each image segment, i.e., co-ordinates_i denotes the coordinate of the wb_i image segment. As shown in Algorithm 1, the working of the function **Pruning** can be broken down into three steps - (i) remove co-ordinates of Noise segment type, (ii) find the normalized area, and aspect-ratios of the bounding boxes, (iii) the area and aspect-ratio of the bounding boxes are compared against the two thresholds: threshold-of-Area (*theta*), and threshold-

of-Aspect-Ratio (*delta*) . They are set to 0.02, and 0.009, respectively∗. As shown in the algorithm, it returns two lists of co-ordinates, Textual co-ordinates and Equation co-ordinates which correspond to the co-ordinates of textual and equation regions in the input image *In*. These coordinate lists are passed on to the next tier to remove textual and equation regions from the input image *In* and generate corresponding masking.

3.3.2 Graphical region segmentation and identification

After extracting the co-ordinates of textual and equation components in the previous module, this tier extracts the co-ordinates of the graphical regions presented in the given document image *In*. Unlike other approaches reported in various existing methods, this study modifies the given input image to convert all the foreground pixels of non-graphical regions into background pixels. The modification of the input image is motivated by the fact that the segmentation model works better when trained on document images where all of the pixels in the foreground belong to the regions of interest of one class. Figure 3.4 shows the schematic diagram of this tier, consisting of four sub-steps: (i) input image masking, (ii) region segmentation, (ii) region classification, and (iii) pruning of false graphical regions.

- *Input image masking:* The goal of this step is to mask the given input image and produce a new input image. The masking is done by converting all the pixels of non-graphical regions to the background pixels. To perform masking, we need two information, namely (i) *co-ordinates of non-graphical regions:* This is fulfilled with the information provided by the previous tier, which are two sets of co-ordinates - Textual co-ordinates and Equation co-ordinates, and (ii) *the background pixels value*

∗The values for both threshold are fine-tune from 0.01 to 0.20, and find 0.02 for *theta* and 0.009 for *delta* provides better result

Algorithm 2 Image_Masking(In, (TC, EC))

//Initialization //
Width, and Height of In as m, and n respectively.
Intensity of the pixel at i^{th} row, and j^{th} column of In as $In(i,j)$
// Determine background pixel intensity //
$t = t.append(In(x,y))$ where $x = \{0$ to $m-1\}$ and $y = \{0$ to $99\}$
$l = l.append(In(x,y))$ where $x = \{0$to $99\}$ and $y = 0ton-1$
$b = b.append(In(x,y))$ where $x = \{0$ to $m-1\}$ and $y = \{(n-100)$ to $n\}$
$r = r.append(In(x,y))$ where $x = \{(m-100)$ to$m\}$ and $y = \{0$ to $n\}$
bp = Highest Frequency intensity in $(t \cup l \cup b \cup r)$
// Pixel value modification //
for each B (bounding box created by the coordinate in $(TC \cup EC)$ **do**
    for each $(i,j) \in B$ **do**
        $In(i,j) = bp$
    end for
end for
Return In

for the given input image - as most of the documents have a common structure that their contents (text, equation, or graphical) starts appearing on the document leaving some blank-regions (background pixels) on all four boundaries, this study calculates the intensity of the background pixels from these four boundaries. The masking process is given by

$$Ig = Image\_Masking(In, \texttt{Textual co-ordinates}, \texttt{Equation co-ordinates}) \tag{3.7}$$

where In is the original input image, and Ig is the modified input image. The working of *Image_Masking*() is demonstrated in Algorithm 2. It can be expressed in two operations:

1. *Determine background pixel intensity* : As stated earlier, this study determines the value of the background pixels based on four borders. The area or

number of pixels to be considered when determining the background pixels value is user-dependent. This study considers 200 rows of pixels (100 each from the top-border and bottom-border of the input document) and 200 columns of pixels (100 each from the left-border and right-border of the input document). From all these pixels, the highest occurring pixel value is determined to be the background pixels value.

2. *Pixel value modification*: The pixel value of all the pixels inside the bounding box obtained with the co-ordinates $h_i \in$ (Textual co-ordinates$\cup$ Equation co-ordinates) are changed with the background pixels value to obtain a new input image *Ig*.

- *Region segmentation :* Given the modified input image *Ig*, same as in the previous tier, the first task is to develop a graphical region segmentation model (we called it as *Graphical-region Segmentation Model (GSM)*). It is developed considering the same UNet architecture as that of ETSM and trained on the document images which have only graphical regions as foreground regions (modifeid input images). GSM is able to capture the characteristics of the graphical regions accurately. Even when the graphical region is made up of sparsely clustered pixels, such as in flowcharts, line charts, and block diagram, it segments the continuous foreground pixel, and is able to segment those graphical regions accurately as a whole. This is the advantage of performing input image masking. It makes GSM to learn the features only from graphical regions without any interference from other components such as textual and equations.

  As shown in Figure 3.4, once GSM produces a mask image, Ig_{mask}, for the given document image *Ig*, the next step is to get the position of the predicted regions of interest. It is done **GetCoordinate** (which we have discussed in Section 3.3.1), and get a set of co-ordinates, $GCoordinate = [g_1, g_2, ...g_p]$ where $g_i =$

$(x_{i1}, y_{i1}, x_{i2}, y_{i2})$. Afterwards, as in first tier for the further process, the model crops and extracts the corresponding segmented images from the input image with **ImageCroping**(discussed in Section 3.3.1), and get a set of segmented images $G = \{g_1.jpg, g_2.jpg.jpg, ...g_p.jpg\}$. This step provides two specific segmentation errors:

- It classifies the collection of some background pixels (blank regions) as the regions of interest. It happens because of some training mask images. When creating training mask images for the graphical components with sparsely clustered pixels, some portions of the regions of interest are filled with continuous background pixels. Due to its inclusion during training GSM, it segments some blank regions as the regions of interest.
- It marks some regions, such as page number, equation number, which are filtered out as false regions of interest in the first tier and hence could not be modified during input image making to become background regions, as the regions of interest of graphical components.

- *Region classification:* The goal of this step is to address the false segmentation caused by the blank regions. A CNN-based model, (which we name it as Graghical-region Classifier (GC) - same architecture as ETNC discussed in Section 3.3.1), is trained to identify the type of image segments ($\in G$) into two classes as demonstrated below:

$$class_i = \begin{cases} \text{Graphical}, & \text{if } GC(g_i) = Graphical \\ \text{Noise}, & Otherwise \end{cases} \tag{3.8}$$

where Graphical denotes segment with graphical components (any regions with foreground pixel), and Noise denotes segment without any foreground pixel,

which is a blank region.

- *Pruning of false graphical regions:* The goal of this step is to address the false segmentation because of the regions that have foreground pixels but are not part of graphical regions. In our case, this error is created by page number, line number, and equation number, which are filtered out from the first tier because of their relatively small area. As shown in Figure 3.4, the pruning process is carried out in **Pruning-G**, which working is same as the **Pruning** (discussed in 3.3.1). The only difference is that **Pruning-G** considers only one parameter, *area of the regions* with a user-defined threshold, threshold-of-Area-graphical (η), to prune falsely segmented regions. This study sets η to 0.03 (The threshold η is fine-tune from 0.01 to 0.06, and find 0.03 provides better result). After pruning, we finally get a new set of co-ordinates, Graphical co-ordinates, which contains the co-ordinates of the graphical regions for the given input image *In*.

3.4 Experimental Set-up

3.4.1 Dataset

This study uses multiple datasets for executing and evaluating the proposed framework.

1. *In-House (IH)* : This study curates a dataset consisting of 4840 document images written in Latin (3840) and Bengali (1000) scripts. In our study, we have observed that Manipuri documents often contain Latin text, including numbers, page numbers, embedded texts, tables, flowcharts, and charts. To address this, we have included documents written in the Latin script in our dataset. This will not only benefit our current research but also aid future efforts in transliterating Manipuri documents and handling code-mixed documents. By considering both the Latin and Bengali scripts, we aim to improve the robustness of our methods

for Manipuri document processing and analysis. All of them consist of graphical components/regions. 4040 samples are considered for training, and the remaining 800 samples are considered for developing a testing dataset. Based on the type of parallel mask images∗, this dataset has two variants:

(a) *In-House-Textual (IH-T)*: Here, the regions of interest are the textual and equation, which means the mask images are annotated in such a way that textual and equation regions are considered as only foreground regions, and the rest as the background regions. This variant is used as the training dataset for ETSM (segmentation model in the first tier). Therefore, it has 4040 samples.

(b) *In-House-All (IH-A)*: In this variant, the masks are annotated to contain the positional and label information of all the regions of interest (equational, textual, and graphical). This variant is used as the testing dataset of the proposed framework. Therefore, it has 800 samples.

2. *In-House-Graphical (IH-Graphical)*: This dataset is the same as IH, but the images are modified manually to have only graphical regions as the foreground regions. Therefore, the document images in this dataset have no textual or equation regions. The corresponding mask images are generated in such a way that all the foreground pixels(which are the graphical regions) belong to the regions of interest. This dataset is used as the training dataset for GSM (segmentation model in the second tier). Therefore, it has 4040 samples.

3. *Text-Equation-Noise dataset (TEND)*: This dataset consists of 6000 samples which can be grouped into three classes of 2000 samples each: Text (images that have only texts), Equation (images that have only mathematical equations), and Noise

∗The parallel mask images are curated manually using a pixel annotation tool *makesense*

(images that does not have texts or equations such as diagrams, and parts of pictures).

4. *Graphical Object-Blank dataset (GOBD)*: This dataset consists of 2700 samples belonging to two classes:"Graphical" (2500 images with a graphical component such as flowchart, table, natural scene images, and sketches.) and "Blank" (200 images without any foreground pixels i.e., images with only background pixels).

5. Four publicly available datasets are used for further performance evaluation, namely *DocBank*[82]∗, *DSSE*[153], *CS*[24] and *Pub*[160]∗

3.4.2 Segmentation and Classification models

The proposed framework consists of two segmentation models (ETSM and GSM), and two classification models (ETNC and GC). As stated earlier, the segmentation models are UNet models with VGG-16 as a backbone of the encoder. Along with vanilla UNet, this study considers three variants of UNet : UNet-CBAM[158], UNet-SE[158], and UNet-(CBAM+SE)[158] obtained with three attention mechanism CBAM[59], SE[59], and CBAM+SE[158], respectively. The experimental setup of the segmentation and classification model for the two tiers are provided below:

- In the first tier of the proposed framework, the segmentation model ETSM is trained using a training dataset of IH-T, while the classification model, ETCM is trained using a training dataset of TEND.

- In the second tier of the proposed framework, the segmentation model GSM is trained on the training dataset of IH-Graphical, while the classification model GC is trained on the training dataset of GOBD.

∗ 1000 samples are used in this study.

Table 3.2: Performance of the proposed framework (with four different UNet variants as ETSM and GSM) against three existing methods with IoU threshold 0.5. mAP: mean average precision, mAF1: mean average F1-score. Bold value indicates the best result.

	Method		Dataset				
			IH-A	**DocBank**	**DSSE**	**CS**	**Pub**
Baseline	Tran, H.T et al.[142]	mAP	94.16	94.92	92.12	89.65	90.02
		mAF1	95.84	93.00	93.04	90.14	92.44
	Wu, X. et al.[151]	mAP	94.31	89.45	93.34	87.31	92.67
		mAF1	87.02	90.04	90.34	88.02	93.91
	Umer, S et al.[144]	mAP	91.53	93.87	90.16	**98.02**	95.34
		mAF1	91.85	94.00	90.50	**98.23**	95.56
Proposed	UNet	mAP	90.21	90.21	89.02	90.01	90.12
		mAF1	90.45	90.01	89.21	91.12	90.87
	UNet-SE	mAP	95.78	97.01	95.21	96.21	95.11
		mAF1	96.01	97.01	95.43	96.56	96.32
	UNet-CBAM	mAP	96.31	97.07	96.34	96.68	97.12
		mAF1	96.78	97.06	96.34	96.94	97.11
	UNet-(SE+CBAM)	mAP	**97.85**	**98.05**	**98.01**	98.01	**97.67**
		mAF1	**97.91**	**97.93**	**98.11**	98.12	**97.87**

3.4.3 Evaluation metric

To evaluate the performance of the proposed framework, this study considers three well known evaluation measures used in various studies[38,116,74,141]: Intersection over Union (IoU) is used for measuring the segmentation performance, mean average precision (mAP), and average F1-score (mAF1) measures are used for object/region classification measures.

3.5 Experimental Results

The proposed framework is evaluated from three different perspective: (i) Evaluation with an optimal IoU threshold value, (ii) Region-wise evaluation: The proposed framework is evaluated tier-wise and step-wise, (iii) Evaluation with varying IoU threshold.

Table 3.3: Segmentation performance of the proposed model against three existing methods over five datasets with IoU threshold 0.5. Performance are measured in mAP (mean average precision).Bold value indicates the best result.

Method	Region	Step	Dataset				
			IH-A	DocBank	DSSE	CS	Pub
Tran, H.T et al.[142]	Text	NA	97.34	95.12	95.87	96.56	98.13
	Eqn.	NA	97.00	94.47	98.17	92.19	97.17
	Graph.	NA	88.00	90.11	81.50	88.69	76.38
Wu, X. et al.[151]	Text	NA	97.87	98.56	94.91	95.78	94.45
	Eqn.	NA	96.58	94.81	94.11	92.65	89.49
	Graph	NA	88.49	84.10	91.61	82.23	89.91
Umer, S et al.[144]	Text	NA	98.112	98.11	97.09	**98.94**	98.34
	Eqn.	NA	96.34	95.19	91.63	98.45	97.97
	Graph.	NA	66.45	61.12	63.68	96.97	76.12
Proposed	Text	1^{st}	86.01	87.45	89.03	89.12	90.12
		$1^{st} + 2^{nd}$	88.12	94.43	91.45	92.19	92.67
		$1^{st} + 2^{nd} + 3^{rd}$	**98.17**	**98.09**	**97.33**	97.56	**98.13**
	Eqn.	1^{st}	81.83	89.34	90.01	90.31	90.11
		$1^{st} + 2^{nd}$	98.12	99.01	99.12	98.12	99.57
		$1^{st} + 2^{nd} + 3^{rd}$	**98.72**	**99.51**	**99.12**	**99.18**	**99.57**
	Graph.	1^{st}	89.21	87.56	82.31	88.45	82.56
		$1^{st} + 2^{nd}$	95.12	95.21	97.65	91.48	97.32
		$1^{st} + 2^{nd} + 3^{rd}$	**97.07**	**97.07**	**98.10**	**98.63**	**98.43**

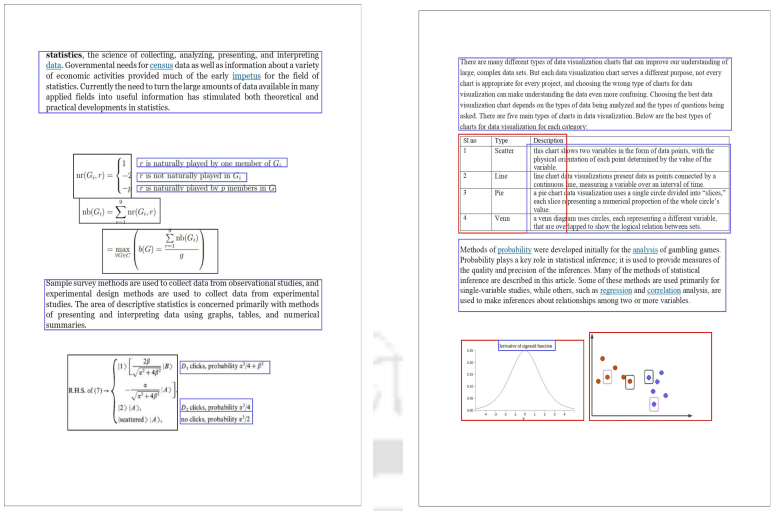
1. *Evaluation with an optimal IoU threshold:* As in various existing studies[38,116,74,141,5,22,127], the segmentation performance is considered optimal with IoU threshold 0.5, this study evaluates the region detection and identification with IoU threshold 0.5. Table 3.2 presents the performance of the proposed framework against three existing methods with IoU threshold 0.5 over 5 datasets namely, *IH-A, DocBank, DSSE, CS, and Pub* . The table shows that the proposed framework with (SE+CBAM) based UNet as the segmentation models (ETSM, and GSM) obtains better detection and identification accuracy for all datasets except for CS. For dataset CS, the existing method[144] exceeds the performance of the proposed framework marginally.

2. *Region-wise evaluation:* In the above evaluation, among the model obtained with the proposed framework, the one with (SE+CBAM)-based UNet as ETSM and GSM provide performance. Therefore, this study considers it as the proposed model and is referred to as our proposed model in the remaining part of this chapter. This study evaluates the performance of the proposed model and three existing methods from the perspective of segmentation and identification of each class/component(textual, equation, and graphical classes in our case) with an optimal IoU threshold of 0.5. The evaluation of the proposed framework follows a specific procedure considering three steps: (i) segmentation using the ETSM/GSM model, (ii) region classification using the ETNC/GC model, where the noisy regions are ignored, and (iii) Filtering out of false regions of interest based on two parameters: area, and aspect ratio, which is the main aim of the *Pruning* process. Firstly, the framework is evaluated after completing the first step, segmentation using the ETSM/GSM model. This evaluation allows us to assess the performance of the framework based on the segmentation results alone. Subsequently, the evaluation is repeated after incorporating the second step, region classification using the ETNC/GC model. This additional evaluation helps us understand the impact of region segmentation followed by region classification on the overall performance of the framework. In this step, all the coordinates of the regions classified as Noise are ignored, and only the regions tagged as textual or equation are retained. Finally, to comprehensively evaluate the framework's effectiveness, we conduct the evaluation after incorporating all three steps, including the last one, in which the false regions of interest are filtered out based on their area and aspect ratio. This final evaluation provides insights into the combined performance of the segmentation, region classification, and pruning steps in achieving accurate positional information for the three classes. The results are

Table 3.4: Performance of the proposed model with varying IoU threshold against three existing methods. mAP: mean average precision, mAF1: mean average F1-score. Bold value indicates the best result.

Method	IoU		Dataset				
			IH-A	**DocBank**	**DSSE**	**CS**	**Pub**
Tran, H.T et al.[142]	0.6	mAP	83.12	87.06	84.05	81.34	82.22
		mAF1	83.24	87.34	84.34	82.44	82.56
	0.7	mAP	75.62	74.63	75.23	74.45	77.52
		mAF1	75.74	75.01	75.24	56.14	77.14
	0.8	mAP	61.72	59.43	62.34	61.05	59.08
		mAF1	62.41	59.58	62.74	61.42	60.06
Wu, X. et al.[151]	0.6	mAP	77.34	75.45	84.23	73.45	85.89
		mAF1	77.89	75.80	85.09	74.02	86.02
	0.7	mAP	69.02	69.12	77.45	69.34	77.47
		mAF1	69.12	69.34	77.35	69.44	77.64
	0.8	mAP	63.12	62.12	67.34	61.05	71.23
		mAF1	64.03	69.23	67.34	60.55	71.44
Umer, S et al.[144]	0.6	mAP	82.32	82.43	82.12	90.12	84.23
		mAF1	82.54	82.67	82.34	90.43	84.44
	0.7	mAP	75.34.	81.76	76.04	82.25	75.07
		mAF1	76.00	81.34	76.19	82.44	75.34
	0.8	mAP	69.45	72.03	69.34	72.52	69.02
		mAF1	69.32	71.78	69.58	72.67	69.34
Proposed	0.6	mAP	**92.25**	**90.00**	**92.01**	**93.01**	**92.67**
		mAF1	**92.31**	**95.03**	**93.07**	**93.02**	**92.04**
	0.7	mAP	90.21	86.12	84.14	86.10	86.56
		mAF1	90.58	86.56	84.84	86.05	84.45
	0.8	mAP	82.15	80.16	80.16	81.01	79.42
		mAF1	82.21	79.23	80.32	81.04	79.28

Figure 3.5: Output samples of the proposed model and three existing methods with IoU threshold 0.6.



(a) (b)

Figure 3.6: Some challenging samples in region segmentation : (a) Equation with multiple options as long text-lines, (b) Graphical component, table with various text-lines as the entries.

reported in Table 3.3. It can be seen from the table that the mAP increases as the step goes on for all three classes. It is further observed that all the models provide acceptable mAP (mean average precision) for two classes, namely Text, and Equation, but the best mAP of the graphical components is provided by the proposed model.

3. *Evaluation with Varying IoU threshold:* We conduct a study on the detection performance of the proposed model with varying IoU threshold. The evaluation is extended with the above three existing methods. Table 3.4 shows the detection statistics of all methods for varying IoU thresholds over five datasets. With the increase in threshold value, it can be seen that the detection accuracy (i.e., the values of mAP and mAF1) for all the methods, reduces. This is because of the reduction in the number of true positive when increasing the IoU threshold. From the experimental results reported in Table 3.2 and this table, Table 3.4, it can

be seen that the best mAP and mAF1, for all the methods (including the proposed model) are obtained when the IoU threshold is set to 0.5. One interesting observation is that the existing method in Study[127] outperforms the proposed framework when IoU threshold value is set to 0.5 (reported in Table 3.2), but it reduces its performance drastically with the increase in IoU threshold. In general, from Table 3.4, it is observed that performance of all the existing methods decreases by increasing IoU threshold value. On the other hand, the performance of the proposed model are comparatively stable while changing IoU threshold value from 0.6 to 0.8. Figure 3.5 presents the output samples of four methods when IoU is set to 0.6. It is observed that other than proposed model, all the methods fail to segment most of the graphical regions accurately. Therefore from the experimenal observation, it can be concluded that the proposed model outperforms existing methods, and it is robust with respect to IoU threshold value.

3.6 Discussion

In spite of providing well behave characteristics compared to various existing methods, our proposed system has some limitations which may be noted in the following points:

- *Textual components:* The proposed framework ignore some details such as page number, line number to be considered as the textual components. In addition to this, it works only on the documents in which the texts are written horizontally.
- *Equation components:* The proposed framework able to segments those equations which are written in a single line accurately. However, it leaves some part of some equation components if they are written in multiple line (if-else) with a text-lines as conditions. This situation is shown in Figure 3.6 (a).

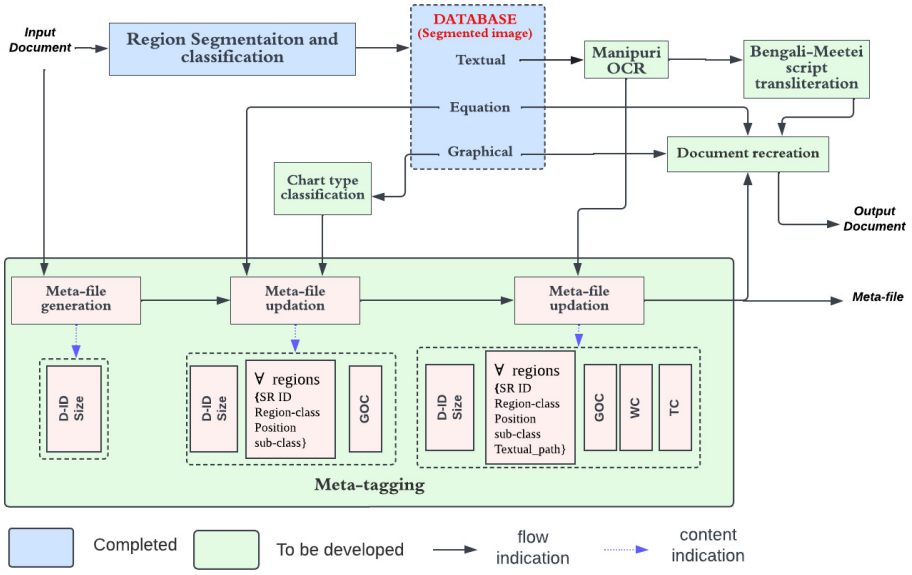


Figure 3.7: Status of the development for the proposed DCS.

- *Graphical components:* The proposed framework fails to segment the complete region of the graphical components if they have long text-line as their parts such as long text-line entries inside the table, and labels of the plots. This situation is shown in Figure 3.6 (b).

- *Background:* This study assumes that the intensities of all the background pixels are same. This assumption does not affect the performance of the proposed framework though.

- *Document layout:* The proposed framework provides better results to the single and the double column documents. If the documents are multiple column (greater than two), the post-processing method in the first phase of the proposed framework filter out multiple regular textual regions considering them as the noisy regions.

3.7 Conclusion and Future work

In this work, we have proposed a 2-tier feedback-based end-to-end integrated framework for document segmentation and region classification. The first tier considers textual and equation regions as the only regions of interest, while the second tier is instructed to consider only the graphical regions as the regions of interest. Through various experimental evaluations against multiple existing methods and datasets, we have observed the advantages of using two tiers and incorporating feedback information. The proposed framework outperforms various existing methods. In the future, we plan to address all the shortcomings discussed in Section 3.6.

3.8 Summary

This chapter presented the proposed method of the module "Region Segmentation and Classification" of DCS. With this module, the input document image is segmented into three regions of interest-*textual, equations*, and *graphical*. The status of the development of DCS can be visualized in Figure 3.7. It can be seen from the figure that the task for the region segmentation and classification is completed, and the segmented images are stored in a DATABASE. The next two chapters of the thesis present the empirical studies and proposed method of the module "Chart Type Classification.".

"If you don't have haters in your way of success, It means you are on wrong way!"

Hassanwainx

4

Chart type classification: Empirical Study

In the previous chapter, the thesis presented the development of a region segmentation and classification model that classifies the true regions of interest for the given document image into three classes - *textual, equation*, and *graphical*. This chapter focuses on graphical components. As mentioned in Chapter 1, DCS aims to classify the graphical components into chart types, the empirical studies of "chart type classification" are

presented herein.

4.1 Introduction

Charts form a powerful tool for visualizing abstract data or scientific findings. Different types of charts have been developed over time to illustrate various types of visualization. Because of its simplicity, accessibility, and intuitiveness, people utilize various charts in a variety of documents. As the number of scientific documents available on the internet with various types of charts increases, automatic chart classification is becoming an important task for various applications (such as information mining, and redesigning) to gain a better understanding of intrinsic information present in the documents. As reported in Section 4.2, there are more than 50 research papers on chart classification with a wide range of methodologies such as model-based, traditional machine learning (ML) classifiers (SVM, KNN, decision tree), deep learning (DL) methods (VGG, Inception, ResNet, Xception, MobileNet). From these studies, we made the following observations:

1. **Dataset Types:** Two types of datasets have been commonly used; real dataset - chart images extracted from real documents, and synthetic dataset - created synthetically using mathematical models.

2. **Small dataset size:** Except for the synthetically generated dataset, most of the existing studies consider small datasets.

3. **Small Number of chart types:** All the existing studies consider a small number of chart types - 14 being the highest and 5 being the most common.

4. **Differences in model performance:** Differences in performance between different models are reported in different studies. For example, the study[67] observes that the deep learning based model, Inception, performs better than AlexNet whereas

study[27] reports the opposite. Another example is that the study[15] presents a traditional classifier, SVM which performs better than KNN, while the opposite is reported in the study[73].

Motivated by the above observations, this work builds 44 different ML models (9 traditional and 35 deep learning) over common evaluation setups. It benchmarks the performances of different chart classification methods to identify their limitations and understand the inherent challenges in chart classification. None of the existing surveys evaluate the methods over the common frameworks and understand the challenges in the chart classification. The remaining part of the chapter focuses on the following points:

- **Common Experimental Setup:** We rebuild all the 44 classification models. The models are evaluated using the following two experimental setups.
  1. *Five folds cross validations:* The performances of 44 ML models are compared under random five-fold validation of a locally curated dataset. The models are further tested over three publicly available datasets[115], [16] and[31].
  2. *With different training sizes:* Different studies have considered datasets of different sizes, and inconsistencies in the reported performances have been observed. To understand the sensitivity of the models to different training sample sizes, we further build the models with different training sample sizes and evaluate their performances.
- **Noisy chart type:** As charts are created for different needs, they come with different types of noises, which may hinder the classification performance. We further identify different types of noise present in the chart and study their effects.
- **Confusing chart pairs:** A classifier may confuse between charts of different characteristics. We further identify charts of confusing nature and their effects.

Table 4.1: The performance of the few selected papers in chart classification that used only real chart datase.

Paper	Year	#Class	Dataset	Model	Accuracy
Zhou et al.[162]	2001	3	840	HNN	86.00
Futrelle et al.[41]	2003	2	129	SVM	91.70
Shao et al.[119]	2006	5	2800	MLP	94.20
Prasad et al.[105]	2007	5	653	SVM	83.80
Huwang et qal.[61]	2007	4	200	Model-based	90.00
Mishchenko et al.[89]	2011	5	980	model-based	89.00
Savva et al.[115]	2011	10	2500	SVM	90.00
Gao et al.[44]	2012	3	300	SVM	97.30
Karthikeyani[72]	2012	4	155	SVM KNN MLP	76.77 78.06 69.68
Amara et al.[2]	2017	11	3377	CNN	89.50
Jung et al.[67]	2017	10	6997	LeNet Inception-V3 AlexNet	44.20 91.30 88.80
Chagas et al.[16]	2017	10	4837	LeNet	70.00
Siegel et al.[124]	2016	7	60000	ResNet-50	86.00
Poco et al.[154]	2017	5	5125	ALexNet	94.00
Dai et al.[27]	2018	5	11174	ResNet-50 Inception-v3 AlexNet	98.89 99.07 99.48
Baji et al.[7]	2019	10	2702	VGG-16	80.00
#Jobin et al.[66]	2019	28	33000	CNN	92.90
Bless et al.[152]	2019	2	2500	VGG-16	96.35
Kreimir et al.[39]	2019	10	2702	VGG-16	89.00
Araújo et al.[16]	2020	13	21099	Xception ResNet-152 VGG-19 MobileNet	95.00 94.60 94.50 94.00

Though the authors in[66] have considered 28 classes, they have also included many of the non-chart classes such as medical images, and natural images. Further, they miss some of the chart types such as , waterfall chart and Reorderable.

4.2 A brief survey

Although the study on chart analysis can be traced back to the year 1991[42], the first study on chart classification is reported in the year 2001[162]. A good survey on chart classification can be found in[107]. Table 4.1 and Table 4.2 present a few of the important selected studies and their details. Based on the classification methods used by the existing studies, the journey of chart classification can be divided into two phases, viz. pre-deep neural network phase (2001-2013) and the deep neural network phase (since 2014).

Table 4.2: The performance of the few selected papers in chart classification that used both real and synthetic chart dataset

Paper	Year	#Class	Dataset	Model	Accuracy
*Balaji et al.[8]	2018	5	6000	MobileNet	99.72
*Chagas et al.[92]	2018	10	Real	VGG-19	60.56
				Inception-v3	76.77
			2683	ResNet-50	77.76
				KNN	38.02
				SVM	45.03
			Synthetic	VGG-19	90.04
				Inception-v3	100
			144871	ResNet-50	99.95
				KNN	97.63
				SVM	99.79
Baji et al.[7]	2019	10	2702	VGG-16	99.55
				VGG-16	80.00
*Davila et al.[31]	2019	7	Real	ResNet-101	88.29
				ResNet-50	77.52
			4242	VGG-19	35.96
				CNN	12.02
			Synthetic	ResNet-101	99.81
				ResNet-50	94.82
			202550	VGG-19	89.78
				CNN	9.59
Araújo et al.[16]	2020	13	21099	VGG-19	94.50
				MobileNet	94.00

*Studies that used only synthetically generated dataset.

In the pre-deep neural network phase, studies exploited model-based approach[61,62,89] and traditional machine learning approach such as SVM[41,115,44], HNN[162], and KNN[44,73]. In a model-based approach, each chart type has its own unique model, which is based on the chart's inherent characteristics. Graphical components of the charts, such as axes and colours[62], the layout of the chart, e.g.,rectangular (for bar charts), circular (for pie charts)[155,89], are among these qualities. The main disadvantage of the model-based approach is that they can only handle predefined charts for which a graphical model exists. Even a minor modification in chart style will be considered as a different model and leads to necessitating creating a new model. As a result, a model-based approach cannot be utilized to classify a wide range of chart styles. Therefore, traditional ML approaches came into play, examining the graphical aspects of charts as a whole object rather than separate parts. SVM, KNN and Decision tree are some well-known state-of-art traditional ML models. They are paired up with various handcrafted

features ranging from visual attributes such as color[48], lines, curves and rectangles[41], basic statistics, entropy and co-occurrence matrix[73], density, distance histogram and profiles[72], to texture features such as the bag of words[115], the local binary pattern[48,139], Hough transform[163,163], and the histogram of oriented gradients(HOG)[105,48,92]. Further, it is observed from Table 4.1, and Table 4.2 that studies in this phase consider manually extracted small sample sizes and the small number of chart types. However, this approach does not generalize well. It is not effective when dealing with a large amount of data that could contain significant varieties, as in the chart image dataset[2]. LeNet[81], AlexNeT[77], VGG-16,19[125], Inception V3[136] and Inception-V4[134] are some of the CNN-based models that have been exploited in the current deep neural network (DNN) phase, along with Inception-ResNet-V2[134], MobileNet-V1[58] and MobileNet-V2[41]. Xception[22] is another CNN-based model that has been exploited in the current DNN phase. Without explicitly extracting features, models take raw images at this phase. Although the authors have investigated models with and without feature extraction in works such as[139,66], they have also commented on the relevance of feature selection approaches. The size of the dataset is one of the obstacles in using DNN models. Most datasets with real chart images are quite small in size. For this reason, the large-scale synthetically created datasets have been examined in a number of research[8,92,31]. However, a model trained on synthetic dataset fails to perform well in real chart images because real chart images often contain noises as compared to synthetic images.

4.3 Experimental Setups

4.3.1 In-house dataset

As mentioned above, most of the earlier studies considered small datasets with a small number of chart types. It may be noted that several new chart types have been used

in recent scientific documents. Many of these chart types have not been considered yet. Motivated by this, we have created a large annotated dataset, which is called as In-house dataset. It consists of 117,271 samples over 28 chart types. We engage two annotators for labeling the samples. These samples are collected from two different sources: (i) Google image search with chart names as queries, and (ii) papers published under CVP∗, ICML†, ACL‡, CHI§, AAAI¶, and ACM‖. The separation of a chart and non-chart from the collection of figures extracted from the documents is done manually. We follow the same procedure provided in the study[66] by using PDFFigure 2.0[23] to extract figures from the documents. We ignore 3D charts, composite charts, a collection of sub-figures, and hand-sketched chart images.

With the above two ways of searching different chart types, we considered only those types in which we could collect at least 1500 samples∗∗. So we ignore some chart types that we found in existing studies and Google searches, such as Interval, Doughnut, and high-low. (because we failed to collect at least 1500 samples for each of them). Our dataset is the largest real chart image dataset with the highest number of chart types, to the best of our knowledge. On the top of In-house dataset, this work considers another three publicly available real chart datasets provided by the studies[115], [16], and[31]. They are considered as test datasets for evaluating the models. As the repositories provided by the above three studies are not maintained, some of the links resolve in error or point to irrelevant images. After curating the samples from the working links and manually verifying the samples, the corresponding datasets of[115], [16] and[32] are named D1, D2, and D3, respectively. The dataset provided by study[32] is the superset of one provided by

∗https://www.journals.elsevier.com/cardiovascular-pathology
†https://icml.cc/
‡https://www.aclweb.org/portal/cljournal
https://dl.acm.org/conference/chi
¶https://aaai.org/Press/Journals/journals.php
‖https://dl.acm.org/journals
∗∗To keep the balance between the sample size, we aim to set the minimum sample size of every chart type as 1500

Table 4.3: Comparison of four datasets. P and SE denote two different sources viz. Publications, and Search Engine, respectively.

Chart Type	Dataset				
	D1	D2	D3*	In-house	
				P	SE
Arc	–	–	–	936	1105
Area	39	246	120	2401	4321
Bar	51	346	429	1251	3757
Box	–	–	316	762	1427
Block	–	–		1872	1140
Bubble	–	–	–	1590	1624
Column	65	340	611	2011	3188
Dendrogram	–	–	–	747	2274
Doughnut	–	–	–	120	1947
Flowchart	–	–	–	891	3210
Gantt	–	–	–	56	2103
Heatmap	–	–	318	650	2569
High-Low	–	–	–	628	1382
Line	85	488	1700	2763	5128
Manhattan	–	–	123	792	1899
Node	–	–	–	1733	4200
Parallel	–	–	–	389	1713
Pareto	63	262	–	738	4683
Pie	27	365	170	2605	2614
Radar	50	50	–	2101	3173
Reorderable	–	–	–	386	1618
Scatter	90	556	2117	2841	5890
Sunburst	–	–		1002	2465
Surface	–	–	110	3002	3440
Table	42	267	–	4034	2507
Treemap	–	–	–	261	2927
Venn	44	343	52	2011	4199
Waterfall	–	–	–	46	2149
Total sample	**320**	**3867**	**3776**	**38619**	**78652**

* D3 is the one provided by [32]. It is the superset of the dataset provided by [31]. Apart from increasing the number of samples to the existing dataset provided by study [31], the study [32] added Heatmap, Manhattan, Venn as new chart types

the study [31]. Table 4.3 compares these three publicly available datasets against In-house. Additionally, examples of each 28 chart type are visualized in Figure 4.1.

4.3.2 Setting up the Classifiers

In pre-deep learning era, SVM, KNN, and Random forest classifiers are commonly used classifiers in chart classification (see Table 4.1). Further, three methods of feature extraction namely Histogram of Oriented Gradients (HOGs), Local Binary Pattern (LBP) and Gray Level Co-Occurrence Matrix (GLCM) are commonly used in chart classifica-

Figure 4.1: Example of 28 chart type

tion[161,105,163,73,48] as well as object detection[29,71]. Considering this, we first evaluate the performance of three classifiers (SVM, KNN, and Random forest) with three different feature extraction methods (HOG, LBP and GLCM). We briefly describe the experimental setups below.

- **Feature Extraction:**
  - *Histogram of Oriented Gradients (HOG):* HOG represents an object using the local distribution of intensity gradients and edge directions. It is the normalized histogram of image gradients with respect to various orientations collected within localized regions in the image. We segment the images into local patches, and extract the features from the patches. We have experimented with various HOG cell sizes of 16×16, 28×28, 32×32, and 64×64. Though study[15] recommends to use 28×28 cell size, an empirical study on our dataset observes that the cell size of 16 x 16 provides the best performance. As a result, HOG related experiments in this work use cell size of 16×16.
  - *Local Binary Pattern (LBP):* In general, LBP encodes local pixel neighborhoods using binary representation. First, we segment the original image resolution 256×256 into 16×16 patches with size $((256/16) \times (256/16))$. Then, in each patch, the LBP feature with radius 2 are extracted and develop a histogram$*$. Finally, a global histogram is created by concatenating histograms obtained from all the patches.
  - *Gray Level Co-Occurrence Matrix (GLCM):* Gray Level Co-Occurrence Matrix (GLCM) extracts features from an image using a co-occurrence matrix. A co-occurrence matrix of an image is a matrix representing distribution of

$*$In study[139], patches of 16 and radius with 2 give the better result, and hence we follow their way.

co-occurring values at a specific offset. The GLCM has been applied in a variety of image analysis applications[33]. As in the study[73], we consider the statistical measure of an image such as area, median, minimum and maximum intensity, contrast, homogeneity, energy, entropy, mean, variance, standard deviation, and correlation, as our features to construct the co-occurrence matrix.

- **Classifiers:** The classifiers SVM, KNN and Random forest (RF) are built considering features extracted using HOG, LBP and GLCM. For KNN, we use 7 as the parameter "k" which is the number of neighbors to be selected.∗. For SVM, the default parameters in Scikit-Learn[102] is used. For random forests, 105 random trees are considered. It has been decided after a brief empirical analysis of values between 50 to 120.

For the deep learning methods, like in various existing studies, we consider the DL models, which are already trained on the ImageNet ISVCR dataset with 1000 classes[98], and fine-tune the models with our In-house dataset for the classification of 28 classes. We consider pretrained AlexNet, Inception-v4, and all 33 different CNN based models provided by keras†, namely VGG -16, 19, ResNet - 50 (v1, v2), 101 (v1, v2), 152 (v1, v2), Inception -v3, Inception - ResNet, Xception, MobileNet - v1, v2 , DenseNet - 121, 169, 201[60], Efficient - B0, B1, B2, B3, B4, B5, B6, B7[137], EfficientNetV2-B0, B1, B2, B3, S, M ,L[138], NASNetLarge[164], and NASNetMobile[164]. In total, this work considers 35 CNN-based models. Except for the EfficientNets, EfficientNetV2s, NASNetLarge and NASNetMobile, all other models have been used in the chart classification task in the literature. As mentioned earlier, we adopt the pre-trained model approach in which the last layer is replaced (because it is the pre-trained model on ImageNet with

∗We tune the value of K ranging from 1 to 10 and finalized to 7 as it gives the best performance.

†https://keras.io/api/applications/

Table 4.4: Mean Accuracy of 9 traditional ML based chart classification models under 5 fold cross validation of In-house dataset. The blue entries highlights the best performance results.

Model	5 fold cross validation			
	D1	D2	D3	In-house
HOG + SVM	74.20 ±0.14	79.49 ±0.15	81.71 ±0.25	85.06 ±0.31
LBP + SVM	68.45 ±0.5	81.06 ±1.21	84.06 ±0.32	80.95 ±0.87
GLCM + SVM	64.08 ±0.32	73.22 ±0.97	79.05 ±0.74	82.39 ±0.54
HOG + KNN	70.91 ±0.38	78.31 ±0.82	80.11 ±0.87	80.84 ±0.97
LBP + KNN	73.05 ±0.17	77.55 ±1.31	79.99 ±0.91	81.80 ±0.87
GLCM + KNN	77.17 ±0.78	77.55 ±0.64	80.54 ±0.68	81.19 ±0.71
HOG + RF	72.81 ±0.52	79.49 ±0.91	80.80 ±0.88	82.76 ±0.82
LBP + RF	67.66 ±0.83	73.60 ±0.54	80.52 ±0.59	82.67 ±0.55
GLCM + RF	69.40 ±0.76	73.64 ±0.67	80.05 ±0.85	82.14 ±0.79

1000 classes) by another dense layer with 28 classes. After that, the last convolutional layers are retrained up to the end. In this way, only the layers more specialized for the specific problem of chart image classification are fine-tuned. We use Stochastic Gradient Descent (SGD) as an optimizer, 0.9 as momentum, 0.0001 as learning rate, 40 as batch-size, and 2 as steps-per-epoch.

4.4 Experimental Observations

4.4.1 Five folds cross validation

Table 4.4 shows average accuracy over five folds cross-validation of SVM, KNN, and Random forest with HOG, LBP, and GLCM feature extractors on the In-house dataset. It also shows the models' performances on the other three publicly available datasets (D1, D2, and D3) as test cases. From the table, we observe the followings.

- All the nine classifiers provide better classification accuracy when tested with an In-house dataset than the testing on the D1, D2, and D3. It may be because of

Table 4.5: Performance of 35 CNN based chart classification models under 5 fold cross validation of In-house dataset. *ma* and *std* denote mean accuracy, and standard deviation respectrively. The blue entries highlights the best performance results.

Model			Testing dataset							
			D1		D2		D3		In-house	
			ma	*std*	*ma*	*std*	*ma*	*std*	*ma*	*std*
VGG	**16**		78.54	±0.59	81.06	±1	81.75	±0.51	89.46	±0.55
	19		78.24	±1.09	81.05	±1.21	84.06	±0.32	88.83	±0.47
ResNet	**50**	**v1**	69.74	±0.76	72.73	±0.56	79.24	±0.56	79.02	±1.01
		v2	69.84	±0.52	73.28	±0.85	80.75	±0.75	78.93	±0.55
	101	**v1**	69.62	±0.3	72.61	±1.03	81.83	±0.47	83.40	±0.53
		v2	70.28	±0.28	74.2	±3.46	81.55	±0.48	83.90	±0.47
	152	**v1**	71.05	±0.39	72.69	±1.03	80	±0.8	83.59	±0.55
		v2	69.04	±0.81	73.65	±0.53	81.45	±0.69	83.44	±0.45
Inception	**v3**		72.37	±0.71	73.03	±0.71	79.82	±0.2	79.89	±0.57
	v4		72.55	±0.6	74.10	±0.15	80.00	±0.41	79.53	±0.94
AlexNet			75.57	±1.39	78.51	±0.56	81.59	±0.58	84.10	±0.94
Xception			85.68	±0.07	86.77	±0.01	87.79	±0.21	90.34	±0.11
Inception -ResNet			74.58	±1.72	79.57	±0.61	80.12	±0.65	81.57	±1.12
MobiolNet	**V1**		76.55	±0.63	78.25	±1.19	80.80	±1.44	88.98	±0.19
	V2		76.34	±0.78	77.94	±1.22	80.28	±0.49	88.87	±0.53
DenseNet	**121**		84.63	±0.74	85.82	±0.65	86.54	±0.47	89.81	±0.25
	169		84.06	±0.63	85.20	±1.14	85.66	±1.05	89.74	±0.8
	201		83.94	±0.63	84.39	±1.21	84.24	±0.73	89.05	±1.7
Efficient	**B0**		58.73	±1.15	55.69	±0.46	85.95	±0.75	64.91	±1.59
	B1		82.09	±0.41	82.84	±0.5	85.95	±0.62	89.05	±0.49
	B2		83.37	±1.95	84.11	±0.04	86.90	±0.46	86.59	±0.43
	B3		85.28	±2.04	84.71	±0.42	87.62	±0.56	89.09	±0.15
	B4		81.01	±0.21	81.67	±0.47	81.93	±0.73	89.54	±0.38
	B5		82.40	±0.46	80.43	±0.55	81.81	±0.71	89.62	±0.37
	B6		82.26	±1.13	81.66	±0.79	81.35	±0.89	89.87	±0.3
	B7		81.64	±0.67	80.92	±0.79	81.28	±0.55	89.56	±0.38
Efficient V2	**B0**		84.49	±0.23	84.68	±1.19	85.48	±0.36	89.23	±0.49
	B1		85.37	±0.69	85.74	±0.58	86.22	±0.16	89.96	±0.17
	B2		81.90	±1.71	83.80	±0.57	85.41	±0.42	89.63	±0.18
	B3		85.12	±0.26	86.08	±0.39	87.12	±0.35	90.03	±0.06
	S		85.44	±0.96	86.18	±0.89	86.81	±0.57	90.15	±0.08
	M		85.36	±0.09	86.09	±0.72	87.16	±0.14	90.12	±0.38
	L		84.72	±0.93	86.62	±0.54	87.60	±1.28	90.05	±0.09
NASNetLarge			85.16	±1.67	86.06	±0.09	86.7	±0.26	90.16	±0.04
NASNetMobile			82.85	±0.44	82.46	±0.57	81.47	±0.89	89.89	±0.22

the fact that the In-house dataset has a smaller percentage of noisy samples as compared to the other three. Noise analysis is discussed in section 4.5.2. It can also be seen that performance corresponds to the percentage of noise present in the dataset, i.e., the In-house dataset has the least noise percentage, then D3, D2,

and D1, respectively.

- In most cases, classifiers with HOG outperform the classifiers with LBP or GLCM.
- Among the classifiers, SVM with HOG outperforms the rest.

Table 4.5 shows average accuracy under five-fold cross-validation of 35 CNN-based models on In-house and further tested on D1, D2, and D3 datasets. The following observations may be noted.

- All 35 classifiers provide the best results with an In-house dataset. In the traditional classification approach, all the models perform poorly over dataset D1 compared to D2 and D3. However, in this case, Efficient B5, B6, and B7 provide better results over D1 than the other two. Most of the models provide better results in the given sequence: In-house, D3, D2, and D1.
- With the In-house dataset, the best-performed models are Xception, Efficient V2-B3, S, M, L, and NASNetLarge. These six models provide a mean accuracy of $\geq$ 90%. Following them, VGG-16, all versions of DenseNet, all variants of EfficientNet (except for B0, and B2), Efficient V2-B0, B1, B2, and NASNetMobile provide a mean accuracy above $\geq$ 89%. Further, both versions of MobileNet and VGG-19 provide mean accuracy above 88%. Finally, other remaining models provide a mean accuracy of less than 85%. It can be seen from the table that all six best-performed models provide consistent performance as denoted by small *std*.
- With Dataset D1, Xception, Efficient-B3, Efficient V2-B3, S, M, NASNetLarge outperforms all other models by accruing a mean accuracy of $\geq$ 85%. Among them, Xception and Efficient V2-M provide more consistent performance with of only $\pm$0.07, and $\pm$0.09, respectively.

- In the case of Dataset D2, with the mean accuracy of ≥ 86%, the best performance comes from seven models: Xception, Efficient V2-B3, S, M, L, NASANetLarge. Among these seven models, Xception and NASNetLarge provide more consistent performance with the σ of only ±0.01 and ±0.09, as shown in the table.

- Finally, with dataset D3, there are three best-performed models, viz: Xception, Efficient V2-B3, M, and L, with a mean accuracy of ≥87%. Efficient V2-M offered more stable performance with a of only ±0.14. Xception follows it with a σ of ±0.21.

- Among all the classifiers, ResNet, Inception, and combined provide the least performance for all the datasets.

- As observed in the above points, Xception and Efficient - V2 B3 are common models for all four datasets that provide high mean accuracy. Among these two models, Xception delivers more consistent performance.

From Table 4.4 and Tabel 4.5, it is evident that except for a few deep learning models like Xception, EfficientNet V2(B3, S, M, L), DenseNet, VGGs, traditional classification set-ups, HOG+SVM, and GLCM+RF also provide comparable performance with the other deep learning models. We have experimented various models with a varied range of classes: 2 classes, 9 classes, 25 classes. The same model changes its characteristics with the change in the number of classes or considered chart types. In this case of 28 classes, Xception and Efficient V2-B3 provide the highest accuracy consistently for all four datasets. In the case of binary classification, VGG-16 outperforms all other bulkier models. This proves that every bulkier model does not outperform small models in every situation. As reported in this work, for almost four datasets, HOG+SVM, GLCM+KNN outperform some CNN - based models like ResNet-50s, ResNet-101s, ResNet-152s, Inceptions, Inception-ResNet, EfficientNet B0. It is further observed that AlexNet, which

Table 4.6: Accuracy of 9 traditional classification set-ups with the increase in the training example.

Model	Training size(# sample)			
	23454	**46908**	**70362**	**92816**
HOG+SVM	79.36	83.97	87.02	87.62
LBP+SVM	68.91	78.01	82.78	78.22
GLCM+SVM	75.67	80.18	80.21	81.34
HOG+KNN	75.64	81.43	81.67	81.96
LBP+KNN	68.08	74.12	82.01	81.52
GLCM+KNN	70.03	76.04	80.59	82.71
HOG+RF	68.01	72.02	77.91	82.45
LBP+RF	70.21	79.32	82.01	81.17
GLCM+RF	81.01	83.98	87.12	84.42

is one of the old state-of-the-arts with straightforward architecture, outperforms very dense CNN such as ResNets, and EfficientNet B0.

4.4.2 Effect of training sample sizes on classification performance

As noted above, different studies have considered datasets of different sizes and reported different performances. To understand the sensitivity of the models, we further build the above models with different training sample sizes. Our dataset is divided into five-folds, and one fold is kept as a testing set. We build four classifiers from the remaining folds by incrementally considering samples in one fold, two folds, three folds, and four folds. These four classifiers are built for all the nine traditional classification setups and the 35 CNN-based models. For the first set of experiments with a single fold, the number of training samples for a single chart type ranges from 400-1600. In average, it has 800 training samples per char type. In the second set of experiments with two folds, another proportionate number of samples are added to the first setup for all the chart types, and so on.

Table 4.6, and Table 4.7 shows the accuracy of all the nine traditional classification setups, and 35 CNN-based models, respectively. The tables show that different classification models provide different convergence characteristics with the increase in training

Table 4.7: Accuracy of 35 CNN based chart classification models with the increase in the training example.

Model	Training size(# sample)			
	23454	46908	70362	92816
AN	71.87	79.56	83.14	88.34
VGG16	67.89	73.98	88.82	88.23
VGG19	71.08	74.87	89.98	89.45
R50v1	67.12	69.14	78.09	74.99
R50v2	68.34	72.91	79.78	77.24
R101v1	71.11	77.91	81.89	83.06
R101v2	74.14	78.12	81.9	83.98
R152v1	77.76	79.12	83.01	83.87
R152v2	78.67	81.04	84.48	84.05
Inv3	68.12	71.67	80.98	79.12
Inv4	72.12	77.04	80.45	79.01
IR	69.21	74.21	79.34	82.16
MNv1	68.112	73.67	87	87.05
MNv2	62.31	89.67	89.12	89.67
Xception	72.9	82.12	91.35	91.57
Dv121	67.42	72.89	89.95	90.48
Dv169	71.01	74.12	86.12	89.67
Dv201	78.34	81.67	89.98	89.99
EB0	62.34	64.22	62.12	63.67
EB1	58.99	66.43	78.98	88.55
EB2	60.23	64.69	75.33	86.32
EB3	64.22	68.89	78.21	89.12
EB4	62.89	64.89	79.34	89.45
EB5	62.89	72.11	89.67	89.9
EB6	59.89	67.12	79.12	88.98
EB7	71.09	79.78	85.98	90.01
Ev2B0	72.22	89.76	89.11	89.99
Ev2B1	69.21	78.34	89.99	90.12
Ev2B2	69.01	79.76	89.34	89.98
Ev2B3	71.11	78.78	88.63	90.12
Ev2S	69.78	73.89	89.11	89.98
Ev2M	73.11	88.9	89.92	91.02
Ev2L	74.11	84.32	90.02	90.57
NNL	69.35	69.11	89.34	90.12
NNM	57.98	64.13	78.98	90.02

size. We observe the following different convergence characteristics:

1. Some models manage to increase the accuracy with the increase in training size. Among the traditional classifier, GLCM + SVM, and GLCM + KNN provide increasing performance with the increase in training size. For the CNN-based models, the same characteristics is observed in the following 18 models: AlexNet (AN), ResNet-101 v1 (R101v1), ResNet - 101 v2 (R101v2), Inception - ResNet

(IR), DensNet-121 (Dv121), DensNet-169 (Dv169), EfficientNet B1 (EB1), EfficientNet B2 (EB2), EfficientNet B3 (EB3), EfficientNet B4 (EB4), EfficientNet B6 (E64), EfficientNet B7 (EB7), EfficientNetV2 B1 (Ev2B1), EfficientNetV2 B3 (Ev2B3), EfficientNetV2 Smaller (Ev2S), EfficientNetV2 Medium (Ev2M), NASNetMobile (NNM), and NASNetLarge (NNL). Hence they might be able to provide more profound performance with the increase in the training samples.

2. It is observed that an increase in the training size does not affect the performance of some models after some instances. HOG + KNN, MobileNet - v2 (MNv2), and Efficient V2 B0 (Ev2B0) converge at the second fold, which means they reached their saturation point with only 46908 samples, and the increase in their training size has no effect afterward. In the same manner, the following two traditional classification set-ups, and ten CNN based models converges at the third fold: (HOG + SVM , HOG + RF), (VGG-16, VGG-19, ResNet-152 v1 (R152v1), ResNet-152 v2 (R152v2), DensNet - 201 (Dv201), Xception, MobileNet v1 (MNv1), EfficientNet B5 (EB5), EfficientNet V2 B2 (Ev2B2), EfficientNet V2 Smaller (Ev2S)). So, they all reached their best-performed version with only 70362 training samples.

3. We observed another situation where the models provide better performance with a lesser number of training samples. This behaviour is observed for the following four traditional classification setups, and five CNN based models: (LBP + SVM , LBP + KNN, LBP + RF and GLCM + RF), (ResNet-50 v1 (R50v1), ResNet-50 v2 (R50v2), Inception v3 (Inv3), and Inception v4 (Inv4), EfficientNet B0 (EB0)). Except for EB0 (which provides the highest accuracy at the second fold), all other models provide the best performance at the third fold but a drop in accuracy is observed at the fourth fold. In other words, we can state that they fail to provide coherent performance with the training size.

It can be inferred from the above observations (considering results provided by both traditional ML and DL models) that an increase in training sample size may not always provide a proportionate improvement in performance. However, an appropriate number of samples should be considered for each chart type to obtain convergence in performance. In our study, most of the classification setups start converging at the third fold. So the appropriate number of samples is about 2400 ($= 800 \times 3$) samples for each chart type in this work.

4.5 Error Analysis

The above sections show that the best classification performance is about 90.34% mean accuracy, which is obtained with Xception over the In-house dataset. The reduction in performances of all the classifiers is also observed with the datasets D1, D2, and D3. To understand the reason for the low performances of the classification models and the reduction of performances over the other three datasets, we perform the following error analysis. Considering *Xception* as one of the most stable classification models with all four datasets, as observed in the above sections, we consider it for the error analysis. The analysis has been performed from two different perspectives :

- *Confusing Chart-pairs:* Charts of similar characteristics may lead to mis-classification.
- *Noise in Chart:* Presence of noise may lead to mis-classification.

4.5.1 confusing chart-pairs and their effects

In order to identify confusing chart-pairs leading to misclassifications, we analyze the distribution of the misclassified samples across all chart class pairs. Table 4.8 shows the average number of classification outputs (over five folds cross-validation, rounded to integer) obtained with Xception model. The rows indicate the input chart types, and

columns indicate the predicted chart types. The diagonal element could be interpreted as the percentage of correct classification for any given sample. Among all, any samples from the Arc chart have 100% of correct classification. It is followed by a Column chart

Table 4.8: Average (of five folds cross validation, rounded to integer) classification outputs over In-house dataset. AC - Arc, AR- Area, BR-Bar, BX -Box, BL-Block, BB-Bubble, CM-Column, DG- Dendrogram, DN- Doughnut, FC-Flowchart, GT-Gantt, HM-Heatman, HL-High-low, LN- Line, MN- Manhattan, ND-Node, PR-Pareto, PL-Parallel, PI-Pie, RD-Radar, RM-Reorderable Matrix, ST-Scatter, SB-Sunburst, SF- Surface plot, TB-Table, TM-Treaamap, VE-Venn, WF-Waterfall.

	AC	AR	BR	BX	BL	BB	CM	DG	DN	FC	GT	HM	HL	LN	MH	PND	PR	PL	PI	RD	RM	ST	SB	SF	TB	TM	VE	WF
AC	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AR	0	86	**6**	0	0	0	0	0	0	0	0	0	0	**5**	0	0	1	1	0	0	1	0	0	0	0	0	0	0
BR	0	**8**	92		0	0	0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BX	0	0	2	77	0	0	0	0	**15**	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	0
BL	0	0	0	0	91	0	0	0	0	**7**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
BB	0	0	0	0	0	80	0	1	2	1	0	0	0	0	0	**9**	0	0	0	0	0	3	0	0	1	0	2	0
CM	0	0	0	2	3	0	96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
DG	0	0	2	3	3	0	0	85	0	3	0	0	0	0	0	0	0	0	0	0	2	1	1	0	0	0	0	0
DN	0	0	4	0	0	0	0	0	92	0	0	0	0	0	0	0	0	0	**5**	0	0	0	0	1	0	0	2	0
FC	0	3	0	0	3	0	0	4	0	89	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
GT	0	0	3	0	0	0	0	0	0	3	84	1	0	0	0	0	0	0	2	0	0	3	0	0	0	2	0	2
HM	0	0	2	0	0	2	0	1	0	1	0	87	0	0	0	1	0	0	0	2	0	0	0	0	0	3	2	0
HL	0	0	0	**6**	0	0	0	0	0	2	0	0	81	4	0	0	1	0	0	1	0	0	1	0	3	1	0	0
LN	0	0	**6**	0	0	0	0	0	0	0	0	0	0	89	0	**5**	0	0	0	0	0	1	0	0	0	0	0	0
MH	0	2	0	0	0	0	1	0	0	0	0	1	0	0	85	0	0	0	0	0	0	**9**	0	0	2	0	0	0
ND	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	87	0	0	0	0	0	**9**	0	0	0	0	1	0
PT	0	0	0	0	0	0	3	0	0	0	0	0	0	2	0	1	86	0	0	1	0	0	1	2	1	0	0	3
PR	0	0	0	0	0	0	3	0	0	0	0	0	0	1	0	1	0	88	0	0	0	0	3	3	0	1	0	0
PI	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	89	0	0	0	2	0	0	0	**7**	0
RD	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	0	3	82	0	0	3	0	0	0	**8**	0
RO	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	87	2	0	0	4	3	0	0
ST	0	0	0	0	0	0	0	0	0	0	0	0	0	**14**	0	0	0	0	0	0	0	83	0	0	3	0	0	0
SB	0	0	2	0	0	0	0	0	0	0	0	1	0	0	2	0	1	0	1	0	0	0	90	0	0	0	3	0
SF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	0	9	0	0	0	0
TB	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	0	0	**7**	0	0	89	0	0	0
TM	0	0	0	0	0	0	2	0	0	0	0	**11**	0	0	0	0		0	0	0	0	0	0	0	0	87	0	0
VE	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	2	0	0	0	2	1	0	0	90	0
WF	0	2	3	3	1	0	2	0	0	0	0	3	0	0	1	2	0	0	0	0	1	3	0	0	0	0	0	79

with a correct classification of 96%, and so on. It can be further stated that among all 28 chart types, classifying box charts correctly is the most challenging, followed by the waterfall chart. If the probability of misclassifying a sample from chart type x as chart type y is above a threshold t , we consider the class pair as confusing chart-pairs. We

have experimented t with 0.02, 0.03, and 0.04. With 0.04(i.e., 4% of misclassification), we are able to find a strong overlapping characteristic among the misclassified samples∗. Considering the 4% threshold, we observe the following 13 confusing chart-pairs from Table 4.8. The chart type-wise error characteristics and their confusing chart-pairs are briefly reported below.

1. **Area chart :** Area charts with multiple regions denoted by parallel or nearly parallel sharp edges are often confused with bar charts. In addition to this, some Area chart samples that have distinct coloured edges but fill up with shaded colour are sometimes classified as line charts. So, with Area chart, we obtained two confusing class pairs: *(Area, Bar)* and *(Area, Line)*.

2. **Box chart:** Some Box chart samples with huge sized multiple boxes are often confused with Dendrogram. So, with Box chart, we have one confusing chart pair : *(Box, Dendrogram)*.

3. **Bubble chart :** The bubble charts with small size bubbles and highly visible background grid are sometimes classified as Node link. From Bubble chart we have one confusing chart-pairs : *(Bubble, Node)*.

4. **Line chart:** Line charts with bigger size of nodes to indicate data points are sometimes confused Node links. Further, some of the Line charts with various coloured backgrounds are sometime identified as Bar charts. So, with Line chart, we observed two confusing chart-pairs : *(Line, Node)* and *(Line, Bar)*.

5. **Manhattan chart:** Manhattan charts with enormous amount of data with not clear edges to indicate vertical margin often classified as Scatter charts. From

∗With 0.02 or 0.03 as t, we fail to confidently draw a similar characteristic among the misclassified samples, instead most of the misclassification are mainly because of noisy samples(which will be discussed in the next section), and lack of training samples.

Table 4.8, it is observed that these two charts are among those charts which are highly confused. So, with Manhattan scatter, we have one confusing chart-pairs : *(Manhattan, Scatter)*.

6. **Node Link:** Some samples of Node links with small nodes but low intensity links are frequently classified as Scatter charts. So, with this chart type, we have one confusing chart-pairs: *(Node, Scatter)*.

7. **Pie chart:** It is observed that Pie charts with one partition dominating other are sometimes classified as Venn. In another case, chart images with multiple pies which have minimum gap between them are also prone to be classified as Venn. So, with Pie chart, we have one confusing chart-pairs: *(Pie, Venn)*.

8. **Radar chart:** Although most Radar charts have hexagonal outer layers, they may have circle or nearly circle-like outer layers. Those samples are often are misclassified as Venn diagrams. So, with this chart type, we only have one confusing chart-pairs : *(Radar, Venn)*.

9. **Scatter Chart:** The scatter charts with lines are sometimes misclassified as line charts. Like the pair (Manhattan, Scatter) these two charts are among those which are highly confused. So, with Scatter chart, we have one confusing chart-pairs : *(Scatter, Line)*.

10. **Table:** Some samples of Table chart without borders and with a crowded data are often classified as Scatter. So, we have one confusing chart-pairs with Table : *(Table, Scatter)*.

11. **Treemap** : From Table 4.8, it is observed that 11% of Treemap samples are classified as Heatmap. They appears to be visually similar most of the time. The main difference is that Treemap has thick or highly visible edges for each blocks.

So, we have another confusing chart-pairs : *(Treemap, Heatmap)*.

A confusing chart class pair (X, Y) denotes that some of the input samples (>= 4% of the testing samples of class X) from chart type X are confused with the chart type Y, and are misclassified as Y. Considering confusing chart pair (X, Y), what we are looking for is the samples $\in X$ classifying as Y, and also the samples $\in X$ misclassifying as Y, which are true positive (TP), and false negative (FN) classification, respectively. However, we cannot consider all the true positive classifications of chart type X as it will include smooth samples of class X, which will get easily classified truly without any confusion with Y. So, we put a criterion to select the TP classification of confusing chart pair (X, Y) from the overall TP classification of chart type X. Let $P(X)$ be the predicted probability of class X $P(Y)$ be the predicted probability of class Y, any truly classified sample $x_i \in X$ will be considered as TP classification if $TP(x_i) = true$, which is defined as

$$TP(x_i) = \left((x_i \in X) \bigcap Condition_1 \bigcap Condition_2\right)$$

where $Condition_1$ is defined as $(P_{x_i}(X) < \partial) \cap (P_{x_i}(X) > P_{x_i}(C))$ for all other classes $C \neq X$, and $Condition_2$ is defined as $(P_{x_i}(Y) < P_{x_i}(X)$ for all classes $Y \neq X$. We further manually verified the TP classification samples of X (with respect to its confusing type Y), $TP_{X,Y}$, to ensure that they have the confusing characteristics to be misclassified as the chart type Y.

Table 4.9 shows the classification outputs of the confusing chart-pairs over the four datasets using Xception. A pair (X, Y) denotes mis-classifying the input samples in chart type x as chart type y. Among all four testing datasets, In-house and D3 contributes more to the confusing chart-pairs as compared to D1 and D2. The CTS in the Table shows the percentage of confusing pairs for a given testing dataset. It is defined as $CTS = \left(\frac{\sum_{(X,Y)}(TP_{X,Y}+FN_{X,Y})}{TS}\right) \times 100$, where $TP_{(X,Y)}$ and $FN_{(X,Y)}$ are the true positives and true

negatives respectively for (X, Y) pair, and TS denotes the entire testing samples. From Table 4.9, it can be seen that In-house dataset contributes to all 13 confusing chart pairs by occupying 7.36% of the whole testing dataset. D3 dataset contributes to only six confusing chart pairs: (Box, Dendrogram), (Line, Bar), (Line , Node) , (Pie , Venn) , (Scatter , Line) , and (Table , Scatter) .These six confusing chart pairs occupy 5.39% of D3 dataset. D2, on the other hand contributes only 3.43% of its population by five types of confusing chart pairs ((Area , Bar) , (Line , Node), (Pie ,Venn) , (Scatter , Line) , (Table ,Scatter)). Among the four dataset, the smallest contribution of confusing chart pairs comes from D1. It contributes only 1.56% of its population by only one type of confusing chart pair which is (Area, Bar).

The TCMC(Total Confusing pairs misclassification) and TCMCO (Total Confusing pairs misclassification overall) rows in Table 4.9 further show the over all error contributions with respect to the confusing pairs, and over the entire dataset. TCMC is estimated as the macro average percentage of sample misclassification between the confusing chart pairs, i.e., $TCMC = \left(\sum_{(X,Y)} \frac{FN_{X,Y}}{TP_{X,Y}+FN_{X,Y}}\right) \times 100$

Similarly, TCMCO is defined by the percentage of misclassifications from the confusing pairs over the entier testing samples (TS), and estimated as below.$TCMCO = \left(\frac{\sum_{(X,Y)} FN_{X,Y}}{TS}\right)$
$\times 100$. From the table, the following points are observed:

- *In-house dataset:* 83.30% of the confusing samples are misclassified. It is further observed that 6.51% of the misclassification (overall) is because of the confusing samples.
- *D1 Dataset):* 80% of the confusing samples are misclassified. It contributes 1.25% of mis-classifications overall.
- *D2 Dataset:* 87.21% of the confusing samples are misclassified. It contributes 3%

Table 4.9: Performance of Xception with respect to the 13 confusing chart-pairs for all four datasets.

Confusing chart pair	**Dataset**							
	In-house		**D1**		**D2**		**D3**	
	TP	**FN**	**TP**	**FN**	**TP**	**FN**	**TP**	**FN**
(Area, Bar)	21	79	1	4	1	21	-	-
(Area,Line)	16	47	0	0	0	0	-	-
(Box, Dendrogram)	11	31	-	-	-	-	0	14
(Bubble,Node)	20	37	-	-	-	-	-	-
(Line,Bar)	11	15	0	0	0	0	2	51
(Line,Node)	9	32	0	0	2	20	2	33
(Manhattan,) Scatter)	5	37	-	-	-	-	-	-
(Node, (Scatter)	21	11	-	-	-	-	-	-
(Pie,Venn)	12	60	0	0	4	6	3	21
(Radar,Venn)	14	56	0	0	0	0	0	0
(Scatter,Line)	21	197	0	0	9	45	4	71
(Table,) (Scatter)	19	34	0	0	1	24	3	42
(Treemap, Heatmap)	11	94	-	-	-	-	-	-
CTS	**7.36**		**1.56**		**3.43**		**5.39**	
TCMC	**83.30**		**80**		**87.21**		**94.3**	
TCMCO	**6.51**		**1.25**		**3**		**5.14**	

of mis-classifications overall.

- *D3 Dataset:* 94.3% of the confusing samples are misclassified. It contributes 5.41 % of mis-classifications overall.

From the above discussion, it is observed that confusing chart-pairs affect the performance of the chart classification model. They contribute more than 5% of the misclassification in case of In-house and D3 datasets, but less than 5% for the datasets D1 and D2. Among the four datasets, both In-house and D3 (in which misclassification of confusing chart-pairs is high) have a higher number of samples per class than the other two. This can infer that with the increase of chart type and inclusion of various samples, the number of confusing chart-pairs is like to be increased.

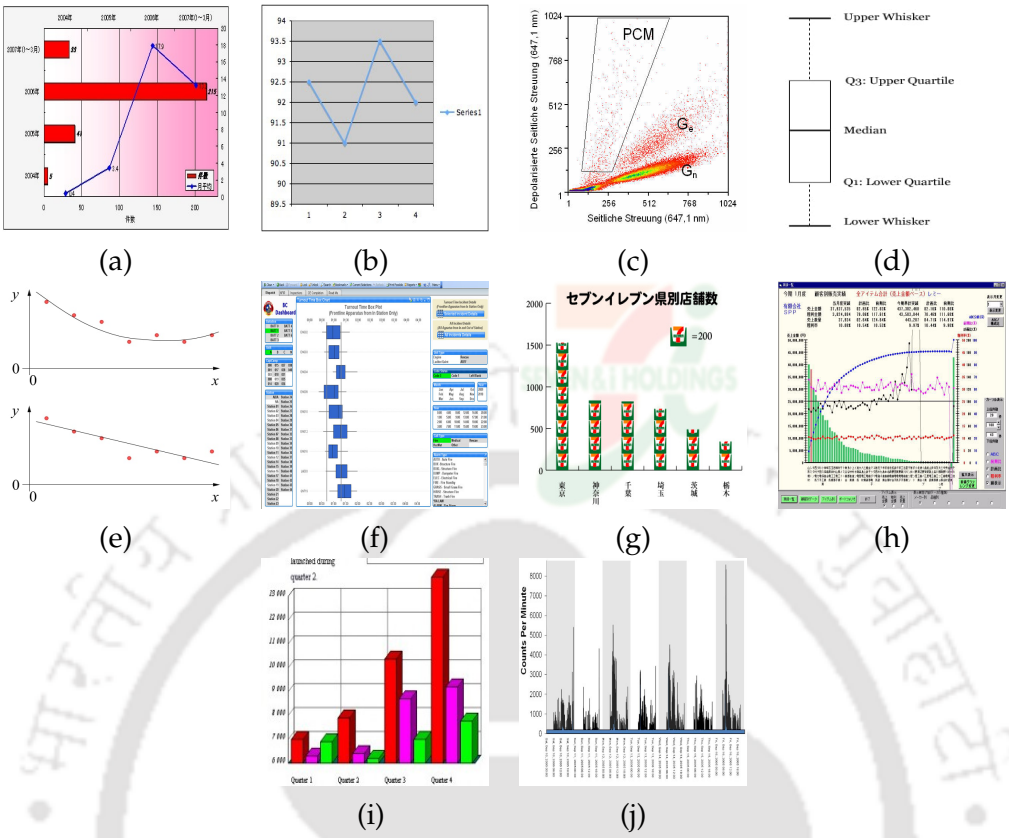


Figure 4.2: Samples of the ten types of chart noise: (a) Composite Chart (CC), (b) Hard BackgroundGgrid (HBG), (c) Additional Information (AI), (d) Text Noise (Text Noise), (e) Transparent Background (TB), (f) Improper Image Screenshot (IIS), (g) Complex Background (CB), (h) Numerous Component (NC), (i) 3D Images (3DI), (j) Patterned Background (PB).

4.5.2 Noise types and their effects

Apart from confusing chart pairs, the mis-classification can also be due to the presence of the noise. In order to identify noise types leading to mis-classifications, we analyse the distribution of the mis-classified samples of all chart types. We observe the following ten noise types. The noise type, error characteristics, and their effects are discussed below.

1. Composite-like chart type (CC): A chart with extra component which resemble

other chart type as shown in Figure 4.2 (a). It is actually a bar chart but composed of bars and two lines.

2. Hard Background Grid (HBG): A chart with hard and dominating background grid lines as shown in Figure 4.2 (b).

3. Additional Information (AI): A chart with embedded information presented in the form of shapes such as circles, rectangles, and traiangle as shown in Figure 4.2 (c).

4. Text Noise (TN) : A chart with enormous amount of additional information presented in the form of text as shown in Figure 4.2 (d).

5. Transparent Background (TB): A chart image with complete transparent background as shown in Figure 4.2 (e).

6. Improper Image Screenshot (IIS):An image with some additional unrelated document regions as shown in Figure 4.2 (f).

7. Complex Background (CB): A chart with distinct background watermark as shown in Figure 4.2 (g).

8. Numerous Components (NC): A chart with multiple chart components such as additional shapes and text as shown in Figure 4.2 (h).

9. 3D images (3DI): As our study do not consider 3D chart images except for surface plot, 3D image become one noise type. Even the image with little degree of the third dimension as shown in Figure 4.2 (i), becomes a noise.

10. Patterned Background (PB): A chart image that have background with patterns such as shown in Figure 4.2 (j). It is an area chart but because of vertical blocks in the background, it is misclassified.

Table 4.10: Performance of Xception with respect to ten noise types over four datasets.

Testing dataset	Noise type																		NTS	TNMC	TNMCO
	CC		HBG		AI		TN		TB		IIS		CB		NC		TN		PB		
	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN	TP	FN	
In-house	0	0	340	181	0	0	0	0	0	0	0	0	0	0	0	0	0	0	172	184	5.2	37.73	3.65
D1	0	3	12	10	2	6	5	11	0	0	3	2	6	8	0	11	0	3	1	14	27.50	63.63	17.5
D2	0	0	38	176	0	79	46	81	0	14	26	4	39	101	0	21	0	5	8	79	18.08	77.6	14.03
D3	6	20	45	95	148	94	11	67	0	9	0	0	14	96	21	190	5	9	16	105	19.64	65.8	12.37

For any given testing dataset, we manually collect the noisy samples considering the characteristics and patterns discussed above. So, for any testing dataset, we form a subset considering only the identified noisy samples. Table 4.10 shows the classification outputs of the noisy samples of the four datasets using Xception. Among all four testing datasets, In - house contributes very less noisy samples as compared to the other remaining datasets. The NTS in the table shows the percentage of noisy samples for a given testing dataset. For any given dataset with N number of noise types, the NTS may be defined as $NTS = \left(\frac{\sum_i^N (TP_i + FN_i)}{TS}\right) \times 100$ where TP_i and FN_i are the true positives and false negatives respectively for the noise type i. In-house dataset contributes to only two types of noise viz. Hard background grid (HGB) and Patterned background (PB) by providing NTS of only 5.2%. Except for the noise type Transparent background (TB), the dataset D1 contributes to all noise types. 27.50% of its dataset is occupied by the noisy samples. Leaving the noise type Composite chart (CC), the dataset D2 contributes to all other remaining nine noise types. Noisy samples from these nine types occupy 18.08% of its dataset. Finally, the dataset provided by D3 occupies 19.64% of its dataset by nine noise types (leaving Improper image screenshot (IIS)).

The TNMC(Total noise misclassification) and TNMCO (Total noise misclassification overall) rows in Table 4.10 further show the over all error contributions with respect to chart noise, and over the entire dataset. TNMC is estimated as the macro average percentage of sample misclassification between the noisy samples i.e.,$TNMC = \left(\sum_i^N \frac{FN_i}{TP_i + FN_i})\right) \times 100$. Similarly, TNMCO is defined by the percentage of misclassifications from the noisy samples over the entire testing samples (TS), and estimated as

$TNMCO = \left(\frac{\sum_i^N FN_i}{TS}\right) \times 100$. From the table, the following points are observed:

- *In-house dataset:* 37.73% of the noisy samples are misclassified. It is further observed that 3.65% of the misclassification (overall) is because of the noisy samples.
- *D1 Dataset):* 63.63% of the noisy samples are misclassified. It contributes 17.5% of mis-classifications overall. It may be noted that not a single instance of CC, and NC types are correctly classified.
- *D2 Dataset:* 77.60% of the noisy samples are misclassified. It contributes 14.03% of mis-classifications overall. It may be noted that not a single instance of AI, TB, and NC types are correctly classified.
- *D3 Dataset:* 65.80% of the noisy samples are misclassified. It contributes 12.37% of mis-classifications overall. It may be noted that not a single instance of TB are correctly classified.

From the above discussion, it is observed that besides confusing chart-pairs, chart noise is another issue to concern while developing a chart classification model. Apart from the In-house dataset, it can be said that the primary cause for the misclassification is noisy samples.

4.6 Conclusion

This work compared nine traditional classification setups and 35 CNN-based models for chart classification. It is observed that the CNN-based model, Xception, dominates all other CNN-based models and traditional classification setups in terms of accuracy and stability. Further, it is observed that, alongside Xception, HOG+SVM, DenseNet-121, and VGG-16 provide consistent performance with respect to the different sizes of the

dataset. We then studied the effect of different sample sizes, the effect of embedded noise in the charts and identified similar chart pairs.

4.7 Summary

This chapter presents the empirical studies of the module "Chart Type Classification" of DCS. The empirical studies focus on the evaluation of multiple stat-of-the-art methods. The study is concluded with the observation of two major challenges in chart type classification. In the next chapter, the thesis presents the development of a chart type classification model which addresses the above two challenges.

"You are the draftsman and craftsman of your own destiny."

Napz Cherub Pellazo

5

Chart Type classification: An Attention-triplet-loss based Chart Type Classification

In the previous chapter, this thesis performs the empirical analysis of the chart type classification, which is one of the key domain of DCS. It discovers two major challenges in the task of chart type classification namely *chart's noise*, and *confusing chart-pairs*.

This chapter presents the proposed an attention and triplet loss based CNN framework to address the above two challenges.

5.1 Introduction

With the increase in the presence of various chart types in scientific documents in electronic media, the development of an automatic chart classification system is becoming an important task. Though the attention of the researchers on chart classification increased post 2001[162], its importance has been realized way back in 1981[42]. Before the deep learning era (pre-deep learning era), initial studies on chart classification mainly employed traditional machine learning methods such as Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Decision Trees, Random Forests, Naïve Bayes, and Logistic Regression, with handcrafted features[119,105,67]. However, the majority of the recent studies focus on using state-of-the-art deep learning models such as VGGs, ResNets, and Xception. As reported in the previous chapter, the majority of the existing chart classification models face problems while handling **(i)Chart noise:** most of the publicly available datasets for chart classification contain samples with various types of noise such as background noise, pattern noise, composite noise, and text noise, and **(ii) confusing chart-pairs:** charts of similar characteristics is also one of the major reason for chart misclassification. To the best of our knowledge, none of the earlier studies on chart classification focused on developing methods that could handle the above two issues. Motivated by this, the study proposes an *attention* and *triplet loss* based model to address the problem of chart noise and confusing chart-pairs. Though attention-based approaches have been extensively used to handle noise in other image classification tasks[146,147], none of the earlier studies have investigated the effect of the attention mechanisms on handling chart noise in the chart classification tasks. Therefore, in this study, we investigate the effect of attention mechanisms, namely Convolutional

Block Attention Module (CBAM)[149] and Squeeze and Excitation network (SE)[59] on handling the chart's noise. We apply these two attention mechanisms to various CNN models (VGGs, ResNets, Inceptions, MobileNets, DenseNets, Xception). From the empirical analysis presented in the previous chapter, it is observed that Xception provides the most effective performance. However, it has not been thoroughly examined with attention mechanisms (except for[157]). In this study, we propose an attention-based Xception model by incorporating CBAM and SE attention with both the residual and non-residual layers (study[157] considers attention only with the last seven residual layers of Xception). Furthermore, this study explores the triplet loss function for the first time in the domain of chart classification. As training a model using the triplet loss function is one of the common approaches for the fine-grained classification[157,145], this study investigates the effect of the triplet loss function on handling confusing chart-pairs classification. The Triplet loss learning method has become a popular approach after the proposal of Facenet[117], created by Google. The goal of the triplet loss is to build a triplet *(anchor, positive, negative)* consisting of an anchor image, a positive image (which is similar to the anchor image), and a negative image (which is dissimilar to the anchor image). Focusing on elongating the distances between confusing samples, we develop a strategy to form the triplets considering only confusing samples.

5.2 Proposed Framework

Since the attention mechanism is one of the popular approaches for classifying fine-grained categories, before developing our proposed framework, this study exploits various classification models with the attention mechanism. Although several studies have introduced attention mechanisms into computer vision, to the best of our knowledge, this is the first of its kind to study the effect of attention in the chart classification domain. As stated in Section 3.1, there are various studies on developing attention-based models

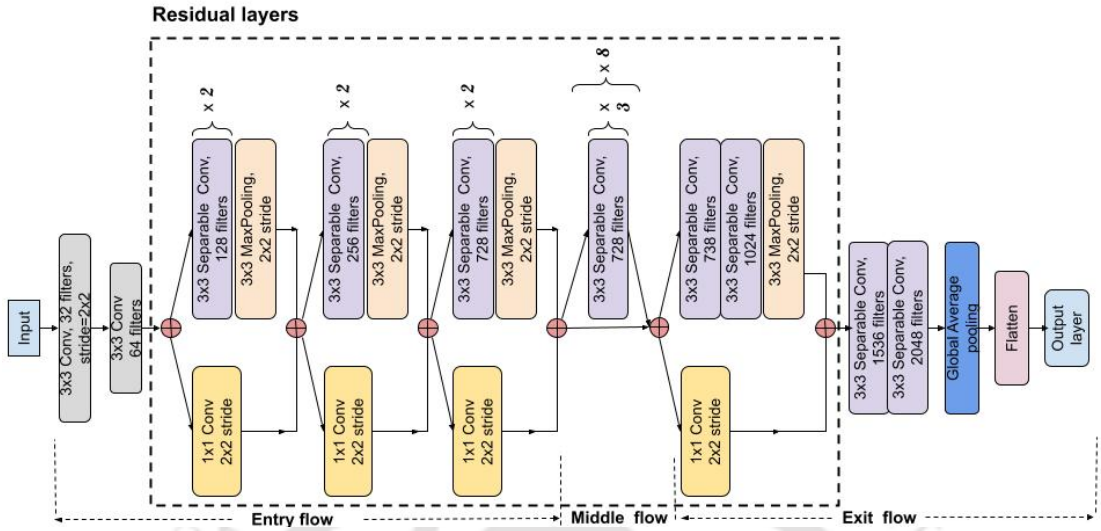


Figure 5.1: Architecture of Xception: 14 modules with 12 residual layers.

of several DL models. However, there is limited exploration of developing attention-based Xception models. Since it is one of the well-performed chart classification models discussed in previous chapter, Chapter 4, we proposed multiple attention-based Xception models and investigated their performance on the chart classification task. This section discusses our proposed attention-based Xception and introduces the proposed framework, which exploits the attention mechanism and triplet loss function.

5.2.1 Attention on Xception

Xception consists of 14 modules with linear residual connections, except for the first and last modules, as shown in Figure 5.1. In other words, it has three main flows: entry (4 modules including the initial CNN layers), middle (8 modules), and exit (2 modules). Study[157] proposed attention-based Xception for the classification of flower types. Their study incorporated the Convolutional Block Attention Module (CBAM)[149] in the last six residual layers. However, we proposed inserting an attention mechanism in the residual layers and the non-residual layers as well. Because of the places where

we can insert attention mechanisms, this study proposed five variants of attention-based Xception :

1. *Xception-Entry (XEN)* - The attention mechanism is inserted only in the entry field. So, in this variant, three attention modules are inserted.

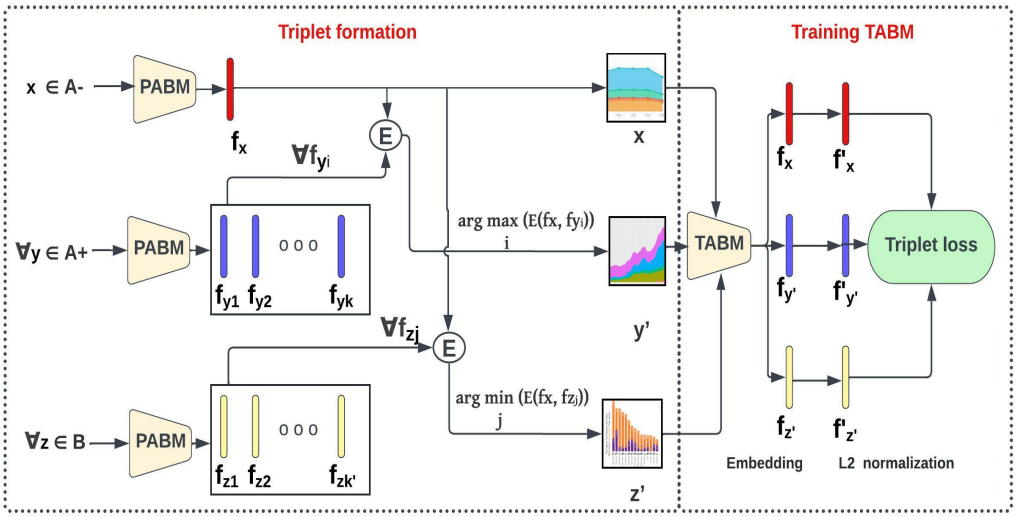
2. *Xception-Middle (XM)* - The attention mechanism is inserted only in the middle field. So, for every eight modules (all of them have a residual connection), one attention module is integrated. Hence, eight attention modules are used in this variant.

3. *Xception-Exit (XEX)*- The attention mechanism is inserted only in the exit field, in which there are two modules (one with a non-residual connection). So, two attention modules are integrated into this variant.

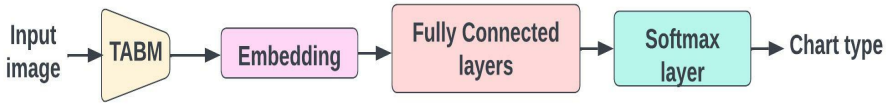
4. *Xception-Middle-Exit (XMEX)*- Considering both flows middle and exit, there are nine residual connections and one non-residual connection. This is the combination of XM, and XEX, where attention modules are inserted into nine residual layers and one non-residual layer.

5. *Xception-All (XA)*- In this variant, all 14 modules of the Xception are followed by the attention module. Hence, 14 attention modules are integrated.

This study considers two well-known attention mechanisms, CBAM and Squeeze and Excitation network (SE)[59]. With five variants of Xception and two attention mechanisms, we have proposed ten attention-based Xception: CBAM-based XE, XM, XEX, XMEX, XA, and SE-based XE, XM, XEX, XMEX, XA.



Block I: Triplet loss training



Block II: Chart type classification

Figure 5.2: Architecture of Proposed framework: Attention-triplet-based chart classification.

5.2.2 Attention & triplet loss based Framework

The schematic architecture of the proposed framework is shown in Figure 5.2. The framework has multi-stage training, which can be broadly divided into two blocks: **Triplet loss training**, and **Chart type classification**.

(1) Triplet loss training

As mentioned earlier triplet loss is first introduced in the study[117] for face verification and recognition in 2015. Since then, it has been one of the popular loss functions for fine-grained classification, such as bio-acoustics[159], species of birds[159], flowers[157], or animals[51] and identifying the models of vehicles[79]. The

goal of triplet loss is to learn parameters by minimizing the intra-class distance and maximizing the inter-class distance as opposed to other loss functions like cross-entropy loss or mean square error loss, where the goal is to learn parameters by minimizing distance between observed and ground truth values. The misclassification because of confusing chart-pairs (R, S) (mentioned in Section 2.1) exists because of the hard-to-distinguish features among the samples of classes R and S. It is observed from the baseline models that the cross entropy loss is not capable of handling confusing chart pairs. However, as triplet loss attempts to capture discriminating characteristics of inter-class and intra-class samples by taking two samples (anchor and positive) from class R and one sample (negative) from class S, it helps in separating the confusing chart pairs of these two classes. So, we adopt the triplet loss function to address the issue of confusing chart-pairs. Its learning can be visualized as shown in Figure 5.3.

In fine-grained classification tasks, apart from the triplet loss, another commonly used method that has the same goal is the contrastive loss[52]. Our study considers triplet loss because of the advantages as opposed to the contrastive loss, as reported in the study[26,51,69,79]. In contrastive loss, even after obtaining the separate clusters of two classes (R and S), there is still a change in the distance between the anchor and positive samples (samples from the same class R) as it tries to put them in the same position. As a result, contrastive loss becomes greedy and cannot tolerate intra-class variance. On the other hand, triplet loss enables clusters to be stretched in order to incorporate outliers while still maintaining a buffer between samples from various clusters. However, triplet loss is computationally more expensive than other commonly used loss functions, such as mean squared error and cross-entropy loss, due to the enormous number of triplets it produces from the dataset. This is one of the major concerns of using

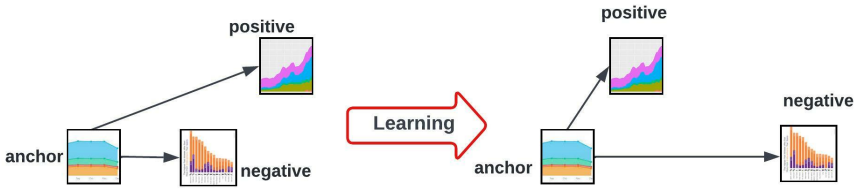


Figure 5.3: The objective of triplet loss learning.

triplet loss for a large dataset. The number of triplets in our analysis, however, is much lower than the potential number of triplets because we already anticipate to be aware of confusing chart pairs.

This block aims to generate triplet samples from the confusing chart-pairs and train the model using the triplet loss function. For a given confusing chart class pair (R, S), the process of this block is described below.

Triplet generation: As stated above, a triplet consists of an anchor sample (a reference sample, which is a confusing sample in our case), a positive sample (which is similar to the anchor), and a negative sample (which is dissimilar to the anchor). Let A be the training set of class R then the set of anchor samples (A^*) , and positive samples (A^+) are defined as $A^* = \{x \in A \mid p(x) \in S\}$, and $A^+ = \{y \in A \mid p(y) \in R\}$, respectively. Where p(.) is the operation that returns a class, it performs a manual checking of patterns in a sample and assigns a class (either R or S) to which the sample might belong. Finally, the set of negative samples is the training set of class S, which is denoted as B. The algorithm for the triplet generation from these three sets is provided in Algorithm 3.

The algorithm consists of two main steps: *Finding feature embedding*, and *Finding hard positive and negative samples*. In the prior step, we used a pretrained attention-based model (PABM) to obtain the anchor's feature vector (f_x), a set of positive feature vectors (denoted by F_{A+}) for all the samples in A^+, and a set of negative feature vectors (denoted by F_B) for all the samples in B. The second step performs

Algorithm 3 Triplet formation

Require: Anchor sample: x, Positive sample's set A^+, and Negative sample's set: B

Ensure: *Triplet* : (x, y', z')

Finding feature embedding:

$f_x = PABM(x), x \in A^*$

$F_{A^+} = \{f_y \mid f_y = PABM(y), y \in A^+\}$

$F_B = \{f_z \mid f_z = PABM(z), z \in B\}$

PABM(.) is the model that is used for feature extraction. PABM stands for Pre-trained Attention-based Model.

Finding the hard positive and the hard negative features:

$f_{xp} = \{f_y \mid f_y = \arg\max E(f_x, F_{A^+})\}$

$f_{xn} = \{f_z \mid f_z = \arg\min E(f_x, F_B)\}$

E(.) is the Euclidean distance between two vectors.

Returning triplet sample:

$y' \leftarrow f_{xp}$ ▷ fetching corresponding image sample $y'(\in A^+)$ of f_{xp}

$z' \leftarrow f_{xn}$ ▷ fetching corresponding image sample $z'(\in B)$ of f_{xn}

return (x, y', z')

distance calculations and comparisons. The Euclidean distance (denoted by $E(.)$) between an anchor feature vector f_x with every negative feature vector (in F_B) and every positive feature vector (in F_{A^+}) are calculated. We have three options to select a positive feature vector $f_{x_p} \in F_{A+}$ and a negative feature vector $f_{x_n} \in F_B$ for a given anchor feature vector f_x : **easy**, **hard**, **and semi-hard**. In an easy selection, the distance between an anchor and a negative is very large than the distance between an anchor and a positive, which can be denoted as $E(f_x, f_{x_n}) >> E(f_x, f_{x_p})$. In the hard selection, the negative feature vector is closer to an anchor than the positive feature vector, which can be denoted as $E(f_x, f_{x_n}) < E(f_x, f_{x_p})$. In the case of semi-hard selection, the negative is not closer to an anchor than the positive but with some margin, which can be denoted as $E(f_x, f_{x_p}) < E(f_x, f_{x_n}) < E(f_x, f_{x_p}) + margin$. As stated in study[54], hard selection yields the best performance. So, we adopt a hard selection process. In the hard selection, two types of masks are identified: *a*

positive hard triplet mask and *a negative hard triplet mask*, to select a hard positive vector f_{x_p}, and hard negative vector f_{x_n}, respectively. f_{x_p} is the one in F_{A^+} which has the highest distance from f_x, and the hard negative sample f_{x_n} is the one in F_B which has a minimum distance from f_x. Finally, for a given confusing chart class pair (R,S), we generate the triplet samples ($x \in A^-, y' \in A^+, z' \in B$) corresponding to the above-obtained triplet embeddings (f_x, f_{x_p}, f_{x_n}).

Training Triplet loss attention based model (TABM): As shown in Figure 5.2, once we have the triplet samples, the next step is to initialize the pre-trained weights of PABM and train with triplet loss to obtain TABM. In the triplet loss function, the idea is to use three identical networks (one each for anchor, positive, and negative) having the same neural net architecture, and they should share underlying weight vectors to train using triplet loss. We implemented this idea using only one network and a triplet, where the network expects three input samples. These three samples do not go with each other but separately. Given a triplet (x, y', z'), in order to estimate the triplet loss, we give x, y' and z' one after another to obtain f_x, $f_{y'}$, and $f_{z'}$ respectively. Once the above-embedded vectors are obtained, as done in [157,145,26], the loss function is estimated using L_2-normalization. The normalized vector $\acute{f}_x$ of f_x is estimated as

$$\acute{f}_{x_i} = \frac{f_{x_i}}{\left(\sum_j f_{x_j}^2\right)^{1/2}} \tag{5.1}$$

Similarly, the normalized vectors $\acute{f}_{y'}$ and $\acute{f}_{z'}$ are also estimated. The distance between the anchor and the positive samples and the distance between the anchor and the negative samples are estimated using Softmax as given below:

$$[S(x, y'), S(x, z')] \leftarrow Softmax\left(E(\acute{f}_x, \acute{f}_{y'}), E(\acute{f}_x, \acute{f}_{z'})\right) \tag{5.2}$$

Table 5.1: Identified number of anchor samples, possible number of negative, and positive samples for each of the 13 confusing chart-pairs.

Sample	Confusing chart class pair												
	Area Bar	Area Line	Box Dendrogram	Bubble Node	Line Bar	Line Node	Manhattan Scatter	Node Scatter	Pie Venn	Radar Venn	Scatter Line	Table Scatter	Treemap Heatmap
Anchor	321	264	119	201	204	176	97	311	132	105	664	374	341
Positive	4534	3331	1463	2075	5735	5939	1736	3930	3454	3655	6321	4412	1190
Negative	4407	6313	2417	4747	4007	4747	6985	6985	4968	4968	6313	6985	2576

$$Loss_t = \frac{1}{|B|} \sum_{(x,y',z')} (S(x,y') + (1 - S(x,z'))) \tag{5.3}$$

We optimize the loss using the Adam algorithm, which is the combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

(2) Chart type classification

In the classification block, the pretrained triplet loss learned model (obtained in the previous block) is used as a feature generator for the final task of chart type classification, as shown in Figure 5.2. It is followed by three fully connected layers and then a softmax layer. The parameters we used in this block are as follows: Stochastic Gradient Descent (SGD) as an optimizer, 0.9 as momentum, 0.0001 as learning rate, 40 as batch-size, and 2 as steps-per-epoch.

5.3 Experimental setup

5.3.1 Dataset

We consider the **In-house** dataset reported in the previous chapter which consists of 110,182 samples with 28 chart types. To perform the triplet formation (described in Algorithm 1), we develop a sub-dataset from **In-house** with the samples from only confusing chart-pairs. The number of anchor, negative, and positive samples for 13

confusing chart-pairs are shown in Table 5.1. With 3308 anchors, this study obtained 3308 triplets. To study the responses of our proposed framework to other datasets, we consider three publicly available datasets, which we have considered while performing empirical analysis reported in the previous chapter: **D1**, **D2**, and **D3**.

5.3.2 Classification model

The classification models considered in this study could be broadly grouped into three: baseline models, attention-based models, and attention-triplet loss based models. They are briefly defined below:

1. *Baseline models :* This study considered 14 DL classification models: VGG (-16, -19), ResNet (-50,-101,-152), MobileNet (-v1, -v2), Inception (-v3, -v4), Inception-ResNet, DenseNet (-121, -169, -201), and Xception. As done in the previous chapter and various studies[67,124,103,8,15], we considered their pre-trained models trained on the ImageNet ISVCR dataset[98].

2. *Attention-based models* : On the top of our proposed five attention-based Xception models and attention-based Xception proposed by the study[157] (it will be referred to as X*), we considered the attention-based models of VGG (-16, -19), ResNet (-50,-101,-152), MobileNet (-v1, -v2), Inception (-v3, -v4), Inception-ResNet, DenseNet (-121, -169, -201), provided by the study[149].

3. *Attention-triplet loss based:* We have considered all the attention-based models used in this study to be trained on using triplet loss as shown in the proposed framework.

Table 5.2: Mean accuracy of 13 baseline CNN based chart classification models under 5 fold cross-validation of In-house dataset.

Testing	VGG		ResNet			Inception		Inception	MobileNet		DenseNet			Xception
Dataset	16	19	50	101	152	v3	v4	ResNet	v1	v2	121	169	201	
In-house	88.98	88.83	79.02	83.4	83.59	79.89	79.53	81.57	88.98	88.87	89.05	88.79	88	**89.91**
D1	78.68	78.248	69.842	70.286	71.05	72.378	72.55	74.58	76.55	76.38	80.63	80.06	79	**80.94**
D2	81.064	81.056	72.732	73.282	72.616	74.2	72.69	79.572	78.258	77.942	81.11	80.2	81.02	**81.39**
D3	81.756	84.06	79.248	81.836	81.552	79.826	80.004	83.12	80.8	80.28	81.05	80.23	80.67	**81.24**

5.4 Experimental results

5.4.1 Baseline Models

Table 5.2 shows the mean accuracy under five fold cross validation of 14 CNN based models on In-house dataset, and further tested on D1, D2 and D3. The following observations may be noted.

- All the 14 models provide the best result In-house, then with D3, D2, and D1, respectively. Xception outperforms all other models.
- Apart from Xception, DenseNets, and VGGs also provide comparatively better performance than the rest.
- Among all the models, ResNet and Inception, provide the least performance for all the datasets.

5.4.2 Attention-based Models

Table 5.3 shows the performance of 19 attention-based models . The following observations may be noted.

- Most of the models experienced a rise in the mean accuracy from that of their baseline version.

- There is a fall in the accuracy from that of the baseline version for all the variants of ResNet in In-house dataset, and Inception-ResNet in D3 dataset.

- For all four datasets, among our proposed five variants of Xception, all the models except for XA improve their performances with an integrated attention mechanism compared to the baseline Xception. With the integration of the attention mechanism on all the modules of Xception, we are retraining all the modules on our dataset, which in turn has no effect of using pre-trained weights. For highly deep networks such as Xception, the size of In-house dataset might not be enough to learn efficiently, and hence XA fails to provide promising results with the attention mechanisms.

- Among all 19 models, XMEX provides the highest mean accuracy for all four datasets.

- Among the two attention mechanisms, all the models provide better results with CBAM for all four datasets.

5.4.3 Attention & Triplet loss based Models

Table 5.4 shows the performance of 19 attention-triplet loss based models, obtained under our proposed framework. The following observations may be noted.

- With our proposed framework, all the models experienced a rise in the accuracy over their respective attention-based models.

- With the proposed framework, XMEX provides better performance for all four datasets.

- Among the two attention mechanisms, our proposed framework works well with CBAM for all four datasets.

Table 5.3: Comparison of 19 attention-based models. The number inside the bracket indicates the rise and fall of the model's accuracy from their baseline versions.

Model	Attention mechanism	Dataset			
		In-house	D1	D2	D3
VGG-16	**CBAM**	90.03 (+1.05)	80.01 (+1.33)	83.23 (+2.166)	83.9 (+2.144)
	SE	89.11 (0.13)	78.9 (+0.22)	82.09 (+1.026)	83.21 (+1.454)
VGG-19	**CBAM**	89.98 (+1.15)	81.89 (+3.642)	82.76 (+1.704)	85.98 (+1.92)
	SE	88.98 (+0.15)	80.45 (+2.202)	82.01 (+0.954)	84.05 (-0.01)
ResNet-50	**CBAM**	75.21 (-3.81)	72.11 (+2.268)	75.67 (+2.938)	82.11 (+2.862)
	SE	74.21 (-4.81)	69.21 (-0.632)	74.01 (+1.278)	86.88 (+8.632)
ResNet-101	**CBAM**	81.98 (-1.42)	72.98 (+2.694)	74.61 (+1.328)	84.09 (+2.254)
	SE	81.01 (-2.39)	74.21 (+3.924)	74.21 (+0.928)	83.51 (+1.674)
ResNet-152	**CBAM**	83.01 (-0.58)	72.01 (+0.96)	75.11 (+2.494)	83.78 (+2.228)
	SE	82.11 (-1.48)	70.18 (-0.87)	74.21 (+1.594)	82.11 (+0.558)
Inception-v3	**CBAM**	83.02 (+3.13)	73.99 (+1.612)	75.67 (+1.47)	84.21 (+4.384)
	SE	81.33 (+1.44)	73.21 (+0.832)	75.02 (+0.82)	82.11 (+2.284)
Inception-v4	**CBAM**	82.11 (+2.58)	74.02 (+1.47)	79.11 (+6.42)	83.99 (+3.986)
	SE	81.12 (+1.59)	73.31 (+0.76)	77.21 (+4.52)	83.32 (+3.316)
Inception-ResNet	**CBAM**	83.12 (+1.55)	78.21 (+3.63)	83.67 (+4.098)	83.12 (+0)
	SE	82.91 (+1.34)	76.45 (+1.87)	81.21 (+1.638)	83.05 (-0.07)
MobileNet-v1	**CBAM**	90.12 (+1.14)	79.89 (+3.34)	82.78 (+4.522)	83.12 (+2.32)
	SE	90.03 (+1.05)	77.45 (+0.9)	81.09 (+2.832)	82.98 (+2.18)
MobileNet-v2	**CBAM**	91.87 (+3)	79.98 (+3.6)	83.22 (+5.278)	82.95 (+2.67)
	SE	91.01 (+2.14)	78.67 (+2.29)	82.17 (+4.228)	82.97 (+2.69)
DenseNet-121	**CBAM**	92 (+2.19)	82.71 (+2.08)	85.21 (+3.318)	84.21 (+1.67)
	SE	90.32 (+0.51)	80.67 (+0.04)	83.9 (+2.008)	83.99 (+1.45)

From the above observations, it is clear that our proposed framework can increase the performance of all the state-of-the-art models. By integrating only the attention mechanism, the models address the issues of noisy charts (discussed in detail in Section 5.5). Yet, the challenges provided by confusing chart-pairs remain unsolved. However, with the combination of attention and triplet loss in our proposed framework, both these issues are addressed on a large scale (discussed in detail in Section 5.5).

Table 5.3: Comparison of 19 attention-based models. The number inside the bracket indicates the rise and fall of the model's accuracy from their baseline versions (Continued).

Model	Attention mechanism	Dataset			
		In-house	D1	D2	D3
DenseNet-169	**CBAM**	91.91 (+3.12)	82.98 (+1.92)	84.78 (+4.578)	84.21 (+2.55)
	SE	89.78 (+0.99)	80.33 (-0.73)	82.67 (+2.468)	83.31 (+1.65)
DenseNet-201	**CBAM**	91.01 (+1.96)	83.06 (+2.12)	85.35 (+3.958)	84.21 (+2.97)
	SE	90.31 (+1.26)	81.09 (+0.15)	83.01 (+1.618)	84.01 (+2.77)
X*	**CBAM**	91.21 (+2.1)	83.45 (+2.5)	87.12 (+5.7)	85.65 (+4.4)
	SE	91.08 (+2.0)	82.11 (+1.1)	86.79 (+5.4)	83.45 (+2.2)
XMEX	**CBAM**	**93.65 (+4.6)**	**87.89 (+6.9)**	**91.89 (+10)**	**87.21 (+5.9)**
	SE	92.89 (+3.8)	83.12 (+2.1)	91.02 (+9.6)	86.31 (+5.0)
XEN	**CBAM**	89.07 (+0.02)	82.13 (+1.1)	81.87 (+0.4)	83.11 (+1.8)
	SE	89.86 (+0.8)	81.21 (+0.2)	80.78 (-0.6)	81.78 (+0.5)
XEX	**CBAM**	91.34 (+2.2)	86.23 (+5.2)	89.89 (+8.5)	85.28 (+4.0)
	SE	92.01 (+2.9)	83.01 (+2.0)	87.36 (+5.9)	82.91 (+1.6)
XM	**CBAM**	91.06 (+2.0)	84.56 (+3.6)	84 (+2.6)	82.76 (+1.5)
	SE	91.23 (+2.1)	82.16 (+1.2)	83.11 (+1.7)	82.45 (+1.2)
XA	**CBAM**	89.01 (-0.04)	82.67 (+1.7)	81.42 (+0.03)	82.01 (+0.7)
	SE	89.99 (+0.9)	81.04 (+0.1)	81 (-0.3)	81.45 (+0.2)

5.5 Discussion

From Table 5.3 and 5.4, it is observed that for all four datasets, triplet loss based CBAM-XMEX (TCBAM-XMEX) outperforms all other models in handling noisy samples and confusing chart-pairs, followed by triplet loss based CBAM-Xception* (TCBAM-X*). So, this study presents the quantitative analysis for these models and their earlier versions before training with triplet loss: CBAM-XMEX, and CBAM-X*, and the baseline Xception. We used the Grad-CAM[118] to do the analysis. Grad-CAM is developed for a visualization approach that calculates the relevance of spatial positions in convolutional layers using gradients. Grad-output CAM's clearly displays attended regions since gradients are calculated with regard to a unique class. We attempt to look at how this

Table 5.4: Comparison of 19 attention-triplet loss based models. The number inside the bracket indicates the rise and fall of the model's accuracy from that of their respective attention-based models.

Model	Attention mechanism	Dataset			
		In-house	D1	D2	D3
VGG-16	**CBAM**	95.78 (+5.75)	89.56 (+9.55)	90.23 (+7)	91.23 (+7.33)
	SE	93.78 (+4.67)	88.45 (+9.55)	90.05 (+7.96)	91 (+7.79)
VGG-19	**CBAM**	95.43 (+5.45)	93.12 (+11.23)	90.45 (+7.69)	91.34 (+5.36)
	SE	95.12 (+6.14)	92.64 (+12.19)	90.65 (+8.64)	90.12 (+6.07)
ResNet-50	**CBAM**	84.34 (+9.13)	82.11 (+10)	86.78 (+11.11)	91.09 (+8.98)
	SE	84.02 (+9.81)	80.23 (+11.02)	83.23 (+9.22)	91.07 (+3.19)
ResNet-101	**CBAM**	90.21 (+8.23)	89.45 (+16.47)	89.55 (+14.94)	91.56 (+7.47)
	SE	90.11 (+9.1)	89.65 (+15.44)	84.56 (+10.35)	89.79 (+6.28)
ResNet-152	**CBAM**	94.12 (+11.11)	89.44 (+17.43)	83.11 (+8)	90.34 (+6.56)
	SE	93.14 (+11.03)	86.32 (+16.14)	80.12 (+5.91)	90.06 (+7.95)
Inception-v3	**CBAM**	89.08 (+6.06)	82.89 (+8.9)	89.45 (+13.78)	90.21 (+6)
	SE	87.42 (+6.09)	81.09 (+7.88)	85.89 (+10.87)	90.45 (+8.34)
Inception-v4	**CBAM**	89.56 (+7.45)	88.56 (+14.54)	91.67 (+12.56)	90.12 (+6.13)
	SE	88.67 (+7.55)	87.78 (+14.47)	91.66 (+14.45)	91.56 (+8.24)
Inception-ResNet	**CBAM**	89.34 (+6.22)	87.56 (+9.35)	91.23 (+7.56)	90.45 (+7.33)
	SE	89.11 (+6.2)	85.23 (+8.78)	89.23 (+8.02)	88.79 (+5.74)
MobileNet-v1	**CBAM**	96.02 (+5.9)	89.45 (+9.56)	91.34 (+8.56)	91.34 (+8.22)
	SE	95.67 (+5.64)	89.11 (+11.66)	90.45 (+9.36)	91.45 (+8.47)
MobileNet-v2	**CBAM**	96.21 (+4.34)	88.78 (+8.8)	90.01 (+6.79)	91.34 (+8.39)
	SE	96.03 (+5.02)	86.45 (+7.78)	89.87 (+7.7)	90.11 (+7.14)
DenseNet-121	**CBAM**	97 (+5)	91.56 (+8.85)	92.35 (+7.14)	91.21 (+7)
	SE	96.43 (+6.11)	91.89 (+11.22)	92.06 (+8.16)	90.334 (+6.34)

network makes excellent use of features by monitoring the areas the network considers crucial for predicting a class. The visualization results are shown in Figure 5.4. For all these challenging input images, as shown in the figure, Xception fails to focus on the regions of interest. With the attention mechanism, it is observed that CBAM-X*, and CBAM-XMEX started to focus on the object's regions for some samples and classified them correctly. However, with the combination of triplet loss and attention mechanism, TCBAM-X*, and TCBAM-XMEX, the issues of most of the challenging samples are

Table 5.4: Comparison of 19 attention-triplet loss based models. The number inside the bracket indicates the rise and fall of the model's accuracy from that of their respective attention-based models (Continued).

Model	Attention mechanism	Dataset			
		In-house	D1	D2	D3
DenseNet-169	**CBAM**	96.89 (+4.98)	93.12 (+10.14)	94.67 (+9.89)	94.67 (+10.46)
	SE	96.01 (+6.23)	93.02 (+12.69)	93.78 (+11.11)	90.05 (+6.74)
DenseNet-201	**CBAM**	96 (+4.99)	92.12(+9.06)	93.23 (+7.88)	91.67 (+7.46)
	SE	95.67 (+5.36)	91.45 (+10.36)	92.56 (+9.55)	89.35 (+5.34)
X*	**CBAM**	95.23 (+4.02)	94 (+10.55)	94.01 (+6.89)	92.01 (+6.36)
	SE	95.04 (+3.96)	93.33 (+11.22)	94 (+7.21)	92.76 (+9.31)
XMEX	**CBAM**	**98.05 (+4.01)**	**94.07 (+6.18)**	**95 (+3.11)**	**95.12 (+7.91)**
	SE	97.21 (+4.32)	93.74 (+10.62)	93 (+1.98)	93.98 (+7.67)
XEN	**CBAM**	96.01 (+6.94)	93.1 (+10.97)	91.98 (+10.11)	91.67 (+8.56)
	SE	94.11 (+4.25)	92.4 (+11.19)	91.11 (+10.33)	90.12 (+8.34)
XEX	**CBAM**	96.12 (+4.78)	91.78 (+5.55)	93.67 (+3.78)	93.11 (+7.83)
	SE	94.67 (+2.66)	90.45 (+7.44)	93.31 (+5.95)	91.02 (+8.11)
XM	**CBAM**	94.21 (+3.15)	91.54 (+6.98)	92.13 (+8.13)	90.34 (+7.58)
	SE	92.11 (+0.88)	90.32 (+8.16)	90.42 (+7.31)	89.32 (+6.87)
XA	**CBAM**	90.23 (+1.22)	89.56 (+6.89)	90.32 (+8.9)	90.56 (+8.55)
	SE	89.45 (+0.06)	88.45 (+7.41)	88.45 (+7.45)	87.45 (+6)

resolved with high classification confidence. We can see that the TCBAM-XMEX network's Grad-CAM masks cover the target object areas better than other approaches. It learns to exploit information in target object regions and aggregate features from them and can decrease the distance of intra-class and increase the distance of inter-class samples. Note that target class scores also increase accordingly. This section presents a detailed discussion of the well-performed attention-based Xception model, TCBAM-XMEX against CBAM-XMEX, CBAM-X*(provided by the study [157]), TCBAM-X*, and the baseline Xception concerning ten noise types and 13 confusing chart-pairs.

5.5.1 Noisy charts

As done in the preceding chapter, for any given testing dataset, we manually collect

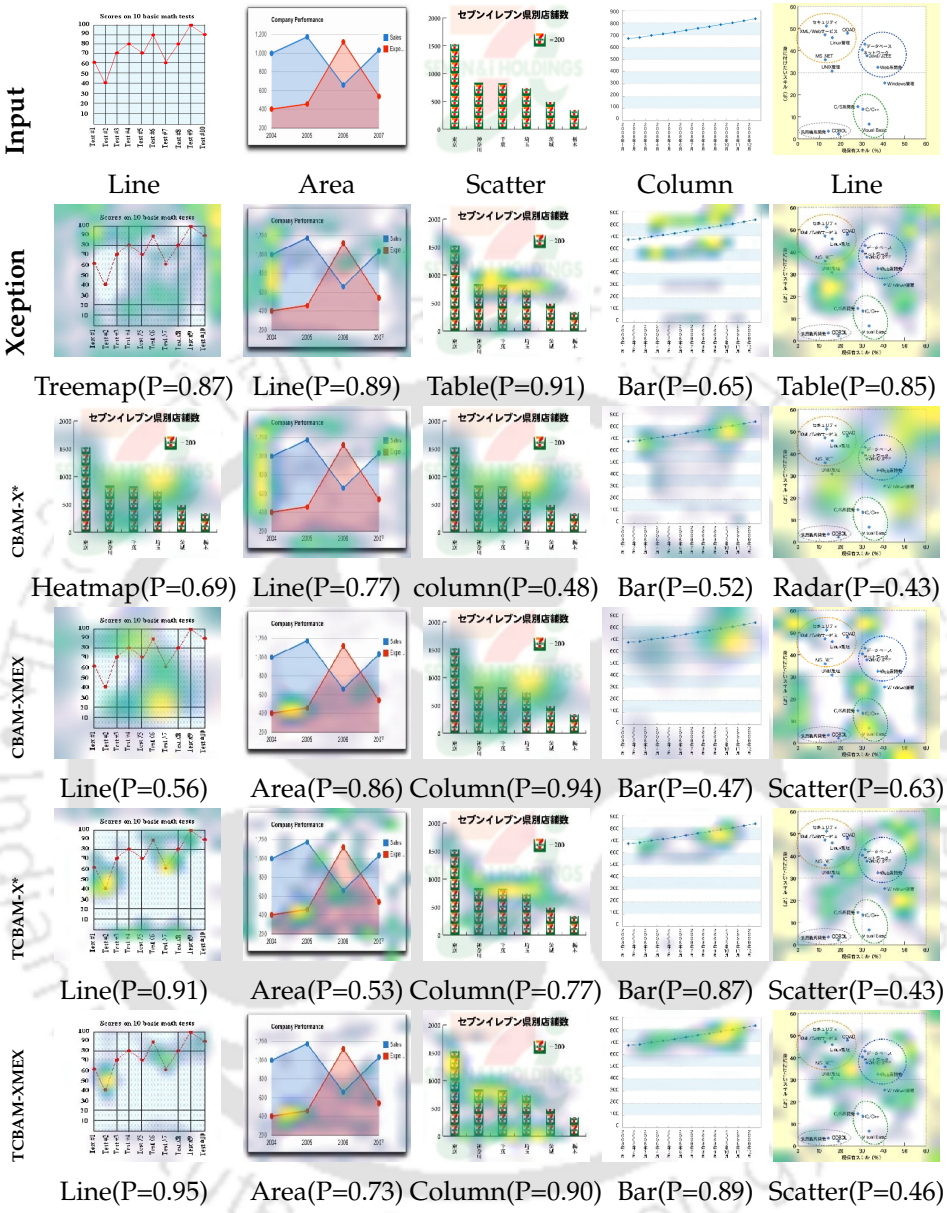


Figure 5.4: Grad-CAM visualization result: Comparison of the visualization results of input image (images in first row), responses from Xception (images in second row), responses from CBAM-X*(images in third row), CBAM-XMEX (images in fourth row), TCBAM-X* (images in fifth row), and TCBAM-XMEX (images in sixth row).The grad-CAM visualization is calculated for the last convolutional outputs. P denotes the softmax score of each network for the classified class.

Table 5.5: Summary of four testing datasets with respect to chart noise.

Dataset	Noise type	# testing samples (TS)	# noisy samples (NS)	% noisy samples in the testing dataset (NTS)
D1	Except for TB, D1 contributes to all	320	88	27.50
D2	Except for CC, D2 contributes to all	3876	701	18.12
D3	Except for IIS, D3 contributes to all	21745	3948	18.32
In-house	HBG and PB	22036	1322	6.00

the noisy samples considering the characteristics and patterns discussed above. So, for any testing dataset, we form a subset considering only the identified noisy samples. Table 5.5 presents the details of four testing datasets with respect to the noisy samples. Among them, In - house contributes very less noisy samples as compared to the other remaining datasets. The NTS in the table shows the percentage of noisy samples for a given testing dataset. For any given testing dataset t_i, the NTS may be defined as $NTS = (\frac{NS_i}{TS_i}) \times 100$, where NS_i and TS_i are the total number of noisy samples, and total number of testing samples, respectively. In-house dataset contributes to only two types of noise viz. Hard Background Grid (HGB) and Patterned Background (PB) by providing NTS of only 6%. Except for the noise type Transparent Background (TB), the dataset D1 contributes to all noise types by occupying 27.50% of its samples. Leaving the noise type Composite Chart (CC), the dataset D2 contributes to all other remaining nine noise types. Noisy samples from these nine types occupy 18.08% of its dataset. Finally, the dataset provided by D3 occupied 18.32% of its dataset by nine noise types (leaving Improper Image Screenshot (IIS)).

Table 5.6 presents the response of Xception, CBAM-X*, CBAM-XMEX, TCBAM-X*, and TCBAM-XMEX on four datasets with respect to the chart noise. The TNMC (Total noise misclassification) and TNMCO (Total noise misclassification overall) column in the table shows the misclassification because of noise among the noisy samples

Table 5.6: Performance of Xception, CBAM-X*, CBAM-XMEX, TCBAM-X*, and TCBAM - XMEX over four datasets with respect to ten types of chart noise. T and F are the true classification and false classification, respectively.

Model	Testing dataset	Noise type																				TNMC	TNMCO
		CC		HBG		AI		TN		TB		IIS		CB		NC		3DI		PB			
		T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F		
Xception	In-house	0	0	514	270	0	0	0	0	0	0	0	0	0	0	0	0	0	0	260	278	**41.61**	**3.65**
	D1	0	3	12	10	2	6	5	11	0	0	3	2	6	8	0	3	0	2	1	14	**63.63**	**17.5**
	D2	0	0	38	176	0	79	46	81	0	16	26	4	39	101	0	5	0	5	8	79	**77.6**	**14.03**
	D3	35	115	256	547	852	121	40	340	0	52	0	0	81	449	29	52	121	449	92	317	**61.8**	**11.22**
CBAM-X*	In-house	0	0	665	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	302	236	**26.49**	**1.67**
	D1	0	3	18	4	4	4	8	8	0	0	3	2	8	6	0	3	0	2	5	10	**47.72**	**13.12**
	D2	0	0	77	137	19	60	58	69	0	16	26	4	39	101	0	5	0	5	17	70	**56.05**	**9.68**
	D3	35	115	343	460	866	107	70	310	0	52	0	0	93	437	34	47	121	449	118	291	**59.03**	**10.72**
CBAM-XMEX	In-house	0	0	720	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	387	151	**16.12**	**1.02**
	D1	0	3	20	2	6	2	13	3	0	0	3	2	10	4	0	3	0	2	7	8	**32.95**	**9.06**
	D2	0	0	98	116	45	24	87	38	0	16	29	1	42	98	0	5	0	5	49	38	**54.21**	**8.02**
	D3	35	115	685	118	903	70	87	293	0	52	0	0	386	144	35	46	126	444	276	133	**17.07**	**4.01**
TCBAM-X*	In-house	0	0	669	116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	311	227	**24.19**	**1.35**
	D1	0	3	20	2	5	1	11	5	0	0	3	2	8	6	0	3	0	2	8	7	**36.32**	**5.91**
	D2	0	0	77	137	19	60	58	69	0	16	26	4	39	101	0	5	0	5	17	70	**27.86**	**5.89**
	D3	35	115	412	391	898	75	90	290	0	52	0	0	111	419	37	44	121	449	210	199	**32.94**	**7.32**
TCBAM-XMEX	In-house	0	0	722	63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	421	117	**13.76**	**0.8**
	D1	0	3	22	0	7	1	12	4	0	0	3	2	12	2	1	2	0	2	12	2	**19.45**	**5.26**
	D2	0	0	202	12	75	4	109	16	0	16	48	9	40	1	99	5	0	5	67	20	**12.55**	**2.27**
	D3	35	115	721	82	911	62	121	259	0	52	0	0	479	51	53	28	126	444	320	89	**13.12**	**3.21**

and over the entire dataset, respectively. TNMC is estimated as the macro average percentage of sample misclassification among the noisy samples i.e., $TNMC = \left(\sum_i^N \frac{F_i}{T_i+F_i}\right) \times 100$, where T_i and F_i are the true positive classification and false negative classification, respectively for the noise type i. Similarly, TNMCO is defined by the percentage of misclassifications from the noisy samples over the entire testing samples (TS), and estimated as $TNMCO = \left(\frac{\sum_i^N F_i}{TS}\right) \times 100$. From the table, the following points are observed:

1. *Xception* : Except for In-house, it provides a false result for more than 50% of noisy samples as given by TNMC. There are some noise types where it recognizes some of their instances, such as PB noise. However, in some cases, it provides inconsistent results by classifying some instances of CC noise correctly (in the case of D3), and sometimes fails to recognize even a single instance of the same

Table 5.7: Summary of four testing datasets with respect to confusing chart-pairs.

Dataset	Confusing chart class pairs	# testing samples (TS)	# Confusing # samples (CS)	% confusing samples in the testing dataset (CCS)
D1	Contributes only to (*Area*, *Bar*)	320	5	1.56
D2	Contributes to **5** confusing pairs (*Area*, *Bar*), (*Line*, *Node*) (*Pie*, *Venn*), (*Scatter*, *Line*) (*Table*, *Scatter*)	3876	133	3.43
D3	Contributes to **6** confusing pairs (*Box*, *Dendrogram*), (*Line*, *Bar*) , (*Line*, *Node*), (*Pie*, *Venn*) (*Scatter*, *Line*), (*Table*, *Scatter*)	21745	1416	6.51
In-house	Contributes to all **13** confusing pairs	22036	1448	6.57

noise type (in the case of D1). The same characteristics are observed for noise type NC, where it fails to recognize a single instance of NC noise in the case of D1 and D2 but provides a true classification for some instances in the case of D3. As given by TNMCO, Xception's performance is highly disturbed by chart noise for all datasets, specifically for D1 and D2.

2. *CBAM-X** : With this attention-based Xception model, an improvement in the performance is observed. TNMC for In-house, D1, D2, and D3 are reduced to 26.49%, 47.74%, 56.05%, and 59.03%, respectively. Even though it provides inconsistent results for some noise types such as NC (like in original Xception), there is an increase in the number of true classifications for other noise types. Like baseline Xception, it fails to classify a single instance of NC noise type for D1 and D2. However, it can correctly classify more NC noise samples from D3.

3. *CBAM-XMEX* : Our proposed CBAM-XMEX model has the same characteristics as CBAM-X*. It does not provide true results for any samples from those noise types where CBAM-X* fails to recognize even a single instance such as CC (for D1, and D2), TB (for D1, D2, and D3), and NC (for D1, and D2). However, it

provides a significant rise in the frequency of correct classifications for other noise types. It classified 386 samples from CB noise type correctly, but CBAM-X* only gets 93. Furthermore, it reduces the TNMC for In-house, D1, D2, and D3 to 16.12%, 32.95%, 54.21%, and 17.07%, respectively.

4. *TCBAM-X** : This model provides promising results compared to the baseline Xception and its version with only the attention module i.e CBAM-X*. It increases the number of true classifications for all four datasets. It reduces TNMC to 24.19%, 36.32%, 27.86%, and 32.94% for In-house, D1, D2, and D3, respectively.

5. *TCBAM-XMEX* : Among these four models, it gives the best performance. Despite the fact that it fails to recognize a single instance of noise types TB and 3DI, the frequency of true classification of all other noise types appears to be increasing. It provides a minor noise error in the case of In-house. It reduces TNMC to 13.76%, 19.45%, 12.55%, and 13.12% for In-house, D1, D2, and D3, respectively.

5.5.2 confusing chart-pairs

Table 5.7 presents the summary of four testing datasets from the view of confusing chart-pairs. It is observed from the table that In-house and D3 contributes comparatively large number of confusing samples than D1, and D2. A pair (x, y) in the table denotes misclassification of the input samples from the chart type $'x'$ as chart type $'y'$. So, the five samples of D1 that contributes to the pair (Area, Bar) are five area chart samples which gets classified as Bar chart type. The CCS in the table shows the percentage of confusing chart samples for a given testing dataset. For any given testing dataset t_i, CCS may be defined as $CCS = (\frac{CS_i}{TS_i}) \times 100$, where CS_i and TS_i are the total number of confusing samples, and total number of testing samples, respectively. As observed in the table, In-house dataset contributes to all 13 confusing chart-pairs providing CCS

of 6.57%, and the lowest CCS of 1.56% comes from D1 that contributes to only one confusing chart class pair.

Table 5.8 presents the performances of five models: Xception, CBAM-X*, CBAM-XMEX, TCBAM-X*, and TCBAM-XMEX over four datasets from the perspective of identified confusing chart-pairs. The TCMC(Total Confusing pairs Misclassification) and TCMCO (Total Confusing pairs Mis-classification Overall) in the table present the overall error contributions among the confusing samples and the entire dataset, respectively. TCMC is estimated as the macro average percentage of sample misclassification between the confusing chart-pairs i.e., the percentage of misclassifications from the confusing pairs over the entire testing samples (TS), and estimated as $TCMC = \left(\sum_{(x,y)} \frac{F_{(x,y)}}{T_{(x,y)}+F_{(x,y)}}\right) \times 100$, where (x, y) is a confusing pair, $T_{(x,y)}$ and $F_{(x,y)}$ are the true positive and false negative classifications, respectively, for (x, y) pair. Similarly, TCMCO is defined as $TCMCO = \left(\frac{\sum_{(x,y)} F_{(x,y)}}{TS}\right) \times 100$. From the table, the following points are observed:

1. *Xception* : It fails to provide promising results for all four datasets. Even though it manages to provide correct classification for some instances of all the confusing chart-pairs for all datastes, it fails to recognize a single instance of the confusing chart-pairs (*Box, Dendrogram*) for the dataset D3. It is further observed that Xception provides TCMC of 94.3% and TCMCO of 6.14% for the dataset D3, which is larger than In-house.

2. *CBAM-X** : This attention based Xception model fails to provide significant performance. It offers some rise in the number of true classification for the confusing chart-pairs (*Area, Bar*) for In-house. However, it yields same characteristics like the baseline Xception for other remaining confusing chart-pairs, for all datsets. Hence ,it fails to deliver significant drop in the rate of misclassification with respect to the confusing samples. It fails to reduce the TCMC for all datasets

Table 5.8: Performance of Xception, CBAM-X*, CBAM-XMEX, TCBAM - X*, and TCBAM - XMEX over four datasets with respect to confusing chart-pairs. *T* and *F* are true and false classificaiotn, respectively.

Model	Testing Dataset	Result	Confusing chart pairs													TCMC	TCMCO
			(Area, Bar)	(Area, Line)	(Box, Dendro)	(Bubble, Node)	(Line, Bar)	(Line, Node)	(Manhat, Scatter)	(Node, Scatter)	(Pie, Venn)	(Radar, Venn)	(Scatter, Line)	(Table, Scatter)	(Treemap, Heatmap)		
Xception	In-house	T	21	16	11	20	11	9	5	21	12	14	21	19	11	**79.26**	**5.21**
		F	125	132	31	37	198	76	37	11	87	56	297	76	94		
	D1	T	1	0	-	-	0	0	-	-	0	0	0	0	-	**80**	**1.25**
		F	4	0	-	-	0	0	-	-	0	0	0	0	-		
	D2	T	1	0	-	-	0	2	-	-	4	0	9	1	-	**87.21**	**3**
		F	21	0	-	-	0	20	-	-	6	0	45	24	-		
	D3	T	-	-	0	-	59	65	-	-	32	0	56	55	-	**94.3**	**6.14**
		F	-	-	267	-	211	201	-	-	142	0	132	196	-		
CBAM-X*	In-house	T	28	16	11	20	11	9	5	21	12	14	21	19	11	**78.83**	**5.16**
		F	118	132	31	37	198	76	37	11	87	56	297	76	94		
	D1	T	1	0	-	-	0	0	-	-	0	0	0	0	-	**80**	**1.25**
		F	4	0	-	-	0	0	-	-	0	0	0	0	-		
	D2	T	1	0	-	-	0	2	-	-	4	0	9	1	-	**87.21**	**3**
		F	21	0	-	-	0	20	-	-	6	0	45	24	-		
	D3	T	-	-	0	-	59	65	-	-	32	0	56	55	-	**94.3**	**6.14**
		F	-	-	267	-	211	201	-	-	142	0	132	196	-		
CBAM-XMEX	In-house	T	45	16	11	20	11	9	5	21	12	14	34	19	11	**72.42**	**4.76**
		F	101	132	31	37	198	76	37	11	87	56	284	76	94		
	D1	T	1	0	-	-	0	0	-	-	0	0	0	0	-	**80**	**1.25**
		F	4	0	-	-	0	0	-	-	0	0	0	0	-		
	D2	T	1	0	-	-	0	2	-	-	4	0	21	1	-	**78.19**	**2.75**
		F	21	0	-	-	0	20	-	-	6	0	33	24	-		
	D3	T	-	-	0	-	59	65	-	-	32	0	92	55	-	**91.86**	**5.98**
		F	-	-	267	-	211	201	-	-	142	0	96	196	-		
TCBAM-X*	In-house	T	137	116	14	28	178	77	24	21	19	21	109	32	100	**36.21**	**2.38**
		F	9	32	28	29	31	11	18	11	13	49	209	63	5		
	D1	T	4	0	-	-	0	0	-	-	0	0	0	0	-	**2**	**1**
		F	1	0	-	-	0	0	-	-	0	0	0	0	-		
	D2	T	19	0	-	-	0	17	-	-	7	0	40	25	-	**18**	**0.6**
		F	3	0	-	-	0	4	-	-	3	0	14	0	-		
	D3	T	-	-	201	-	245	212	-	-	109	0	127	217	-	**21.53**	**1.4**
		F	-	-	66	-	25	54	-	-	65	0	61	34	-		

except for In-house (reduced by only 0.43%).

Table 5.8: Performance of Xception, CBAM-X*, CBAM-XMEX, TCBAM - X*, and TCBAM - XMEX over four datasets with respect to confusing chart-pairs. *T* and *F* are true classification and false classifications, respectively (continued).

Model	Testing Dataset	Result	Confusing chart pairs													TCMC	TCMCO
			(Area, Bar)	(Area, Line)	(Box, Dendro)	(Bubble, Node)	(Line, Bar)	(Line, Node)	(Manhat, Scatter)	(Node, Scatter)	(Pie, Venn)	(Radar, Venn)	(Scatter, Line)	(Table, Scatter)	(Treemap, Heatmap)		
TCBAM-XMEX	In-house	T	144	148	38	56	209	85	40	32	98	66	315	93	103	**1.45**	**0.09**
		F	2	0	4	1	0	0	2	0	1	4	3	2	2		
	D1	T	5	0	-	-	0	0	-	-	0	0	0	0	-	**0**	**0**
		F	0	0	-	-	0	0	-	-	0	0	0	0	-		
	D2	T	22	0	-	-	0	17	-	-	7	0	40	25	-	**15.78**	**0.54**
		F	0	0	-	-	0	4	-	-	3	0	14	0	-		
	D3	T	-	-	266	-	265	261	-	-	172	0	179	248	-	**1.76**	**0.11**
		F	-	-	1	-	5	5	-	-	2	0	9	3	-		

3. *CBAM-XMEX* : Our proposed CBAM-based Xception, CBAM-XMEX, able to provide marginally better performance than Xception, and CBAM-X*. There is a rise in the frequency of correct classification for the confusing chart class pair (*Area, Bar*) for In-house. Unlike, CBAM-X*. it produces true classification for some of the instances of the pair (*Scatter, line*) for D2, D3, and In-house. Furthermore, it reduces TCMC for In-house, D1, D2, and D3, to 72.42%, 80%, 78.19%, and 91.86%, respectively

4. *TCBAM-X** : As compared to Xception, and CBAM-X*, it provides profound performance not only with noise charts, but also with confusing chart-pairs. It classified all instances of the pair (*Table, Scatter*) for D2 correctly, and provides better results for all four datasets. Unlike the previous model, it able to reduce the misclassification error (because of confusing samples) for all four datasets. The TCMC for In-house, D1, D2, and D3, are reduced to 36.21%, 2%, 18%, and 21.53%, respectively.

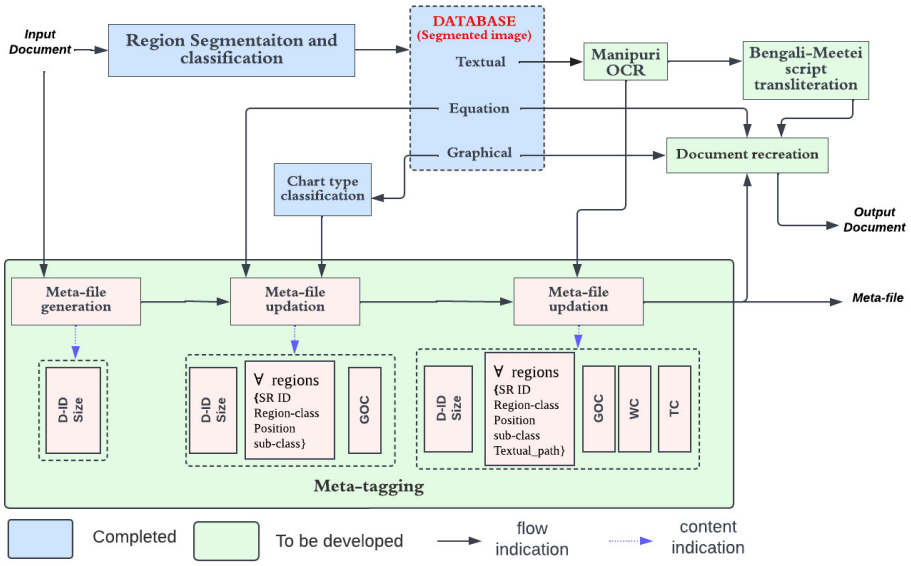


Figure 5.5: Status of the development for the proposed framework.

5. *TCBAM-XMEX* : This model has a significant performance as compared to the above three models. For all four datasets, it increases the number of true classifications of all confusing chart-pairs. Multiple confusing chart-pairs with 100% correct classifications are reported. It's also worth noting that all the confusing samples of D1 are correctly categorised, implying that the contribution to the misclassification due to confused samples is zero. Considering other remaining dataets, In-house, D2, and D3, the TCMC are reduced to 1.451%, 15.78% and 1.76%, respectively.

From the above discussion, it is observed that the attention-based Xception models are capable of addressing some of the challenges presented by noisy data. However, attention-triplet loss based Xception models are able to address both the issues of chart classification: chart noise, and confusing chart-pairs.

5.6 Conclusion

This research offered a framework for dealing with two major chart classification issues: *chart noise* and *confusing class chart pairs*. This is the first study of its kind to tackle these complex and challenging issues in developing the chart classification models. For the first time in the domain of chart classification, the proposed framework used two attention mechanisms, CBAM and SE, as well as the triplet loss function. In addition, the developed framework employed the offline model for producing triplet samples from confusing chart pairs. This study conducted comprehensive trials with multiple state-of-the-art models to evaluate its efficacy, confirming that our proposed framework outperforms all baselines on four different datasets. In addition, we visualize how it infers an input image precisely. Interestingly, we discovered that our framework focuses appropriately on the target object. In a nutshell, the attention mechanism deals with the majority of chart noise, while the triplet loss function tackles the problem of confusing chart pairs. In the future, we intend to expand the number of chart kinds and include their 3D images.

5.7 Summary

In this chapter, the thesis presented the proposed method of the module "Chart Type Classification" of DCS. With this module, the segmented graphical regions are labeled as one of the 28 chart types. The non-chart type graphical regions are labeled as "other". After the development of this module, the status of the development of DCS can be visualized in Figure 5.5. It can be seen from the figure that two major tasks are completed: (i) region segmentation and classification, and saving of segmented images into DATABASE, and (ii) chart type classification. The next chapter presents the proposed model of Manipuri OCR.

"We must value our independence in the same manners
that we value our lives."

Mwanandeke Kindembo

6

Development of OCR System for Manipuri by Adaptation of OCR Systems for Languages with Comparable Scripts

For any given document image, DCS segments it into three regions of interest- *textual, equation* and *graphical*, which was discussed in Chapter 3. Following the segmentation, the graphical components are classified into chart types, which is the second task of

the DCS and was discussed in Chapters 4 and 5. This chapter considers the textual components and presents the development of the Manipuri OCR system, which is one of the key modules of DCS.

6.1 Introduction

The objective of this chapter is to create a robust OCR system for the Manipuri language by leveraging existing OCR systems designed for languages with similar writing scripts. Chapter 1 (Introduction) of this thesis provides comprehensive information about the Manipuri language and its historical background. Therefore, a detailed discussion of these aspects will not be repeated here. However, it is important to highlight the current OCR status of the Manipuri language. OCR has more than 50 years of history, and apart from just developing it, more advanced studies have been made. Currently, most of the reviews on the OCR system focus on enhancing the already developed models from different perspectives. It includes *making it independent of fonts* [5,68,1,106,120] ,*work effectively irrespective of the input image quality* [10,13,154,91,130], *segmentation free model* [111,92,108,65], and *more robust using hybrid classification methods*, and [108,64,12,94,122]. Despite all these advancements, there are minimal studies on building a Manipuri OCR system to the best of the author's knowledge. The probable reason for this lack of research in developing the Manipuri OCR system is the limited availability of the resource. However, as stated above, having one robust Manipuri OCR system is a must, but making it from scratch is an expensive task. Therefore, developing a capable Manipuri OCR system using the limited available resources is an objective of this study.

One possible solution is to adapt the OCR system for Bengali or Assamese, as they have similar scripts. One may be tempted to use the OCR system for Bengali or Assamese on the Manipuri text as its script is identical to their scripts. As Manipuri's language family is different from that of Assamese and Bengali, their language structures

Table 6.1: Distinct characters among the scripts used for Bengali, Assamese, and Manipuri languages.

Language	Frequency(use)	Character									
		Individual					Combine				
		র	ৰ	ৱ	ঙ	ড়	রু	ৰু	র্	র্	ৱু
Bengali	Rare				✓						
	Regular	✓				✓	✓				
	Not used		✓	✓				✓	✓	✓	✓
Assamese	Rare										
	Regular		✓	✓	✓	✓		✓			
	Not used	✓					✓		✓	✓	✓
Manipuri	Rare					✓					
	Regular	✓		✓	✓		✓		✓	✓	✓
	Not used		✓					✓			

are different. In addition to this, a few differences in their scripts contribute to lowering the accuracy on a big scale. Bengali script has 11 vowels (swarabarna), 39 consonants (banjanbarna), and 10 numerals. The distinct characters in the scripts used for these three languages are shown in Table 6.1. A consonant character র (ra) in Bengali and Manipuri is written as ৰ (ra) in Assamese. The consonant ৱ (wa) is generally not used in Bengali but is widely used in Assamese and Manipuri. These are the two most prominent distinctions among the scripts of these languages. The other distinction is due to the combination of dependent vowels with these consonants. Fortunately, they do not form any conjunct characters. Hence distinct combined characters created by these two characters are only those shown in Table 6.1. Another consonant ঙ (nga) is rarely used in Bengali but used widely in Assamese and Manipuri. Finally, the consonant ড় is rarely used in Manipuri as compared to the other two. So an OCR system for any of the above three languages fails to work efficiently for the other two languages. The detailed explanation is done in Section 6.3.

This work proposes three adaptation strategies. The first one explores on enhancing the size of the training text corpus so that the model could accrue reliable performance.

The investigation is done by populating the corpus size with random words in a linear manner. The second strategy investigates how to acquire effective performance by populating the corpus size with a limited number of selective words. The selection of words is carried out after evaluating the baseline OCR model's error on Manipuri documents. This means that the model is trained to learn and understand those symbols that are not seen earlier or seen inadequately. The final strategy combines the above two. It investigates the impact of random populating of training texts after the model has learned those symbols that are not seen earlier or are inadequately seen. This work further perform extended experiment of semi-supervised training to automatically populate the training text corpus.

In summary, this work has contributed to the following:

- Systematic evaluation of the Assamese and Bengali OCR systems on Manipuri text.
- Adaptation strategies to develop a new OCR system from the existing OCR systems.
- Manipuri OCR systems obtained by the adaptation of Assamese and Bengali OCR.
- Semi-supervised training approach : It aims to automatically populate the training text size with unlabeled samples.
- Evaluation of OCR systems : The evaluation of OCR systems involves assessing their performance over diverse datasets sourced from various origins and featuring a wide range of fonts.

6.2 Related work

As this work deals with the development of OCR for a low-resource Manipuri language, the survey is done in two stages. Firstly, the survey is done on the development of an OCR system for the low-resource languages and secondly, Manipuri OCR systems for Bengali script are reviewed.

6.2.1 Research work on the development of the OCR system for low-resource languages

State of the arts on the development of OCR systems for the low resource languages can be divided into two groups and described below.

1. Adaptation of the existing OCR system

   Ghosh et al.[47] developed an Assamese OCR system using the Bengali OCR system. The adaptation was possible because of the similarity of the scripts use for Assamese and Bengali. Their research considered the Bengali OCR system developed by[18] as the base model for the adaptation. The classification for the baseline Bengali OCR system is done in two stages. Firstly, the characters with similar shapes are pooled into one group, and the final stage recognizes an individual character from the pool. To develop the Assamese OCR system, the modification was done in both arenas. Their study first finds out the classified group from the first stage where the extra Assamese characters can be added, and after adding, the second stage's classifiers got retrained. The classification accuracy of the Assamese script increased only by modifying in the second stage. Still, the modification was done in the first stage too, and the efficiency increased from 90% to 99%.

2. Development of OCR system for low-resource language using Tesseract OCR en-

gine

Since the Tesseract OCR engine became an open source in 2005, many studies started to use its training features to recognize the scripts of other languages. The initial versions of the Tesseract OCR are based on traditional computer vision algorithms. However, with its latest version, 4.0, and 5.0, it has implemented a Long Short Term Memory (LSTM) network. Since its version 4.0, it works well with 127 official languages and 37 scripts. Apart from computerized document pages, various studies[112,4,140,150] utilized Tesseract OCR for various images, such as license plate images and complex natural scene images. As Tesseract OCR Engine provides a systematic procedure to develop an OCR for any language, various studies utilized it to develop their own OCR. Here, we present a discussion of some selected studies that adopt the Tesseract OCR engine for the development of OCR of a low-resource language.

White et al.[148] trained the Tesseract OCR engine (version 3.01) to support Ancient Greek. Their work pointed out the different areas for the Tesseract to improve in non-English languages. The challenges faced by the Tesseract in recognition of ancient Greek script were the diacritics characters. This is fairly a general issue because many scripts, including Bengali, have diacritics. Mamata Nayak and Kumar[95] developed an OCR system for printed documents of the Odia language used in the state of Odisha, India. Their work retrained the Tesseract-based English OCR on a set of Odia characters. As the Tesseract generates box files only by considering the nature of the English character, it failed to produce box information for Odia's character correctly. Therefore, their work performed manual checking of box files and created a bounding box of each Odia character by using the algorithm used in[148]. Ibrahim et.al.[63] developed OCR for the Dhivehi language, which is spoken in the South Asian island country of the Maldives,

Table 6.2: Some samples of Manipuri words predicted by the Tesseract-based Assamese and the Bengali OCR system.

Expected	Predicted by Bengali OCR	Predicted by Assamese OCR
তৌরকথিবা	তৌরকথিবা	তৌৰকথিবা
ৱারী	রারী	ৱাৰী
কৌথোক্লগাা	কৌথোক্কগা	কোথোক্লগা
খঙদোক্তুনা	খণদোল্লুনা	খণদোল্লুবা
মখোয়দা	মখোয়দা	মখ্যয়দা
এরিয়া	এরিয়া	এবিয়া
ৱাটরশেদ	দ্বাটরশেদ	দ্বাটৰশেন
চথখি	চণ্থখি	চথখি
ইম্ফাল	ইনল্ফাল	ইম্লাল
থৌবাল	থৌবান	যৌবাল
র্ৱক	র্বক	র্ৱক

written in the script called “Thaana”. Their work used Tesseract version 3.01 and was able to moderate performance. Unfortunately, the model cannot be generalized for commercial or industrial use because of its low performance. The main reason for low performance is stated as the core nature of the Tesseract, i.e., since it was primarily designed to identify English text, introducing new languages to the system brings along its own set of challenges. Each new language may have unique characteristics, phonetic variations, and script conventions that need to be considered and accommodated in the transliteration process. In another study by Udawatta et al.[86], Tesseract OCR Engine was trained to recognize the Sinhala script. It is used for the Sinhala language, which is one of the low-resource languages with a native speakers of around 3 million. Their work used around 20 examples of each character to train. By working on improving the image quality, their work claimed that the trained model could recognize Sinhala characters effectively.

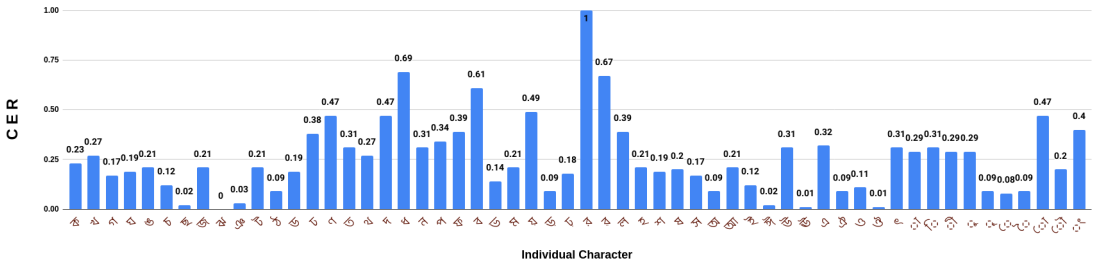


Figure 6.1: Character-level Error Rate (CER)[ranging from 0 to 1] of 56 individual characters provided by the Assamese OCR system on Manipuri text.

6.2.2 Research work on the development of Manipuri OCR system for the Bengali script

Chaudhuri and Pal[17] proposed an OCR system to read two Indian scripts: Bengali and Devanagari. Their proposed system performed satisfactorily on single-font clear documents. The author reported that the system can be used for Assamese and Manipuri. However, they did not provide any evaluation results. In another study by Mathew et al.[87], they proposed a multilingual OCR system for Indic scripts. Their research considered 12 Indian languages, including Manipuri. They considered Manipuri only because it uses almost the same script as Bengali.

6.2.3 Evaluation metrics

In this chapter, as stated earlier, our focus is on the development of Manipuri OCR through an adaptation approach from existing OCR systems. An important aspect of this development is the selection of appropriate evaluation metrics to analyze the performance and characteristics of existing OCR systems that deal with similar scripts. Therefore, this section provides an overview of the evaluation metrics that have been adopted for our study. We used *Character Level Accuracy (CLA)* and *Character-level Error Rate (CER)* to assess the performance of the OCR models. The *CER* for a specific

character α, CER_α is defined as the ratio of the total α's C_{error}, C_{error_α} to the total number of its occurrences, (C_{R_α}) in the reference text. Mathematically, it can be expressed as:

$$CER_\alpha = C_{error_\alpha} / C_{R_\alpha} \tag{6.1}$$

C_{error_α} can be defined as

$$C_{error_\alpha} = \sum_{n=1}^{N} I((P(n) \neq R(n)) \cap (R(n) = \alpha)) \tag{6.2}$$

where $P(n)$ represents the predicted character at position n in the text, $R(n)$ represents the reference character at position n in the text, N represents the total number of characters in the reference text, and the function $I(.) = 1$ if $((P(n) \neq R(n)) \cap (R(n) =' \alpha'))$ is **True**, otherwise, **0**. The CER_α provides a measure of the error rate specifically for the character α, taking into account the total number of errors made in predicting that character relative to its frequency in the reference text. This study utilizes the OCR-evaluation tools∗ to obtain the C_{error} . In the same manner, the Character Level Accuracy (CLA) for a given character α can be defined as

$$CLA_\alpha = (C_{correct_\alpha} / C_{R_\alpha}) \times 100\% \tag{6.3}$$

where $C_{correct_\alpha}$ is the number of correctly predicted α, which be defined as

$$C_{correct_\alpha} = \sum_{n=1}^{N} J(P(n) = R(n) = \alpha) \tag{6.4}$$

where where $P(n)$ represents the predicted character at position n in the text, $R(n)$ represents the reference character at position n in the text, N represents the total number of characters in the reference text, and the function $J(.) = 1$ if $((P(n) = R(n)) = \alpha))$ is **True**

∗https://github.com/Shreeshrii/ocr-evaluation-tools

otherwise **0** .

6.3 Evaluation of the Assamese and the Bengali OCR system on Manipuri Documents

This section discusses the suitability of using the Assamese and Bengali OCR systems to recognize texts from Manipuri documents. The Assamese and the Bengali OCR system provided by the Tesseract OCR engine (version 5.0) are considered to analyze the applicability. The Character-level Error Rate (CER)∗ [ranging from 0 to 1] is calculated for only 56 individual(vowels, consonants, vowel diacritics) and 307 conjunct characters, commonly used in Manipuri. For this task, we created a testing image generated from our text dataset termed as Manipuri Test Document(MTD) which consists of 29 pages with 14,700 words, 29,124 individual and 21,375 conjunct characters. The texts are extracted from Naharolgi Thoudang†, a local news daily printed in Manipuri. The extracted texts are saved as PDF files and then converted to image files. The conversion of PDF to image file, in this study, is done using imagemagick with Linux command convert . The mentioned news daily is considered as our source of text corpus because it is the only Manipuri news publication in UTF-8. Besides this MTD dataset, we developed three more main datasets which will be discussed in section 6.5.1.

6.3.1 Evaluation of the Assamese OCR system on Manipuri documents

As mentioned above, this study uses the Assamese OCR system provided by the Tesseract engine as it is one of the effective open-source engines. Figure 6.1 and Table 6.3 shows the CER produced by the Assamese OCR system for the individual and commonly used conjunct characters in Manipuri documents, respectively. Figure 6.1 shows

∗The metric we considered for the evaluation of OCR is discussed in Section 5.2.
†http://naharolgithoudang.in/web/

that the character র has the highest CER with 1 (i.e, all the occurrences of র in Manipuri documents are mis-classified), followed by ধ with CER of 0.69 and so on. The character ঈ has the lowest CER with 0.02. It is observed from the figure that there are errors in recognizing every character with an average error rate of 0.2. Table 6.2 further shows a few output examples for the Assamese OCR system on Manipuri documents. From Figure 6.1 and Table 6.2, the following are noted.

- The Manipuri র (ra) (the *ra* in Bengali script) is not used in Assamese. As a result, the character র (ra) in a Manipuri document is always recognized as one of the three similar characters (ব (Ba) , ৰ (Ra) and ৱ (Wa)). Among these three characters, the prediction rate of র as ৰ is the highest. Table 6.2 shows that the character র in ৰাটৰশেদ is recognized as ৰ, while in এৰিয়া, it is recognized as ব .
- Characters with similar structures are sometimes recognized interchangeably. Table 6.2, shows that the character খ in তৌৰকখিবা is recognized as থ, and character থ in থৌবাল is recognized as য.

Because of the error in recognizing the character র , the associated combined characters like র্র,রু have the highest CER with a value of 1. Other combined characters that have CER of value 1 are the combined characters with ৱ . Table 6.3 shows the distribution of CER in recognizing conjunct characters. The average CER of recognizing conjunct characters is 0.31. Most of the errors in conjunct characters are because of their complicated and similar structures. It is shown in Table 6.2 that the character ক্ল in কৌথোক্লগা is recognised as ক্ল and ম্ফ in ইম্ফাল is recognized as ল্ফ.

Contributed by all the above factors, Assamese OCR provides Character level accuracy (CLA)∗ of 66.56% on the MTD dataset. Out of 50,499 characters, 16,877 are misclassified by the Assamese OCR system.

Table 6.3: Character-level Error Rate (CER) provided by the **Assamese OCR system** for the 307 commonly used Manipuri conjunct characters. Column *Ci* presents conjunct characters and their respective CER.

C1		C2		C3		C4		C5		C6		C7		C8	
ষ্ঠ	0.9	শ্ব	0.5	ঙ্ঘ	0.4	ম্ড	0.3	স্চ	0.3	ঘ্র	0.2	স্ঢ	0.2	ষ্ত	0.1
জ্ঝ	0.9	শ্ধ	0.5	ন্ধ	0.4	চ৵	0.3	ল্ল	0.3	গ্ম	0.2	স্ত	0.2	চ্য	0.1
ধ্র	0.9	চ্র	0.5	ঢ৵	0.4	চ্চ	0.3	ণ্ভ	0.3	চ্ন	0.2	ণ্ঢ	0.2	ঘ্য	0.1
ঞ্চ	0.8	ধ্ম	0.5	ন্দ	0.4	ফ্র	0.3	ণ্ন	0.3	ম্ণ	0.2	ণ্ম	0.2	জ্ম	0.1
জ্ঝ	0.8	জ্ঞ	0.5	স্ধ	0.4	ল্জ	0.3	ল্ষ	0.3	ঘ্ন	0.2	ণ্প	0.2	ট্য	0.1
জ্জ্ম	0.8	প্ঞ	0.5	ত্ত	0.4	শ্ফ	0.3	ত্ব	0.3	ষ্ট্র	0.2	য৵	0.2	ন্দ	0.1
ম্ঞ	0.8	ঙ্গ	0.5	জ্র	0.4	প্ছ	0.3	গ্র	0.3	ট্র	0.2	ঙ্ম	0.2	ঢ্য	0.1
ক্ষ	0.8	ষ্ট্র	0.5	ণ্ফ	0.4	শ্প	0.3	স্ল	0.3	প্চ	0.2	ল্ল	0.2	ষ্ম	0.1
ঙ৵	0.8	ট্র	0.5	ণ্শ	0.4	ল্ট	0.3	ল্হ	0.3	ন্থ	0.2	ত্ন	0.2	প্থ	0.1
ল্ঞ	0.8	ল্থ	0.5	ভ্ল	0.4	প্ড	0.3	ম্ট	0.3	ল্ণ	0.2	ণ্স	0.2	শ্শ	0.1
ল্ছ	0.8	শ্ষ	0.5	প্শ	0.4	জ্ণ	0.3	শ্গ	0.3	ম্ধ	0.2	প্ম	0.2	ম্ল	0.1
ছ্র	0.8	ণ্ড	0.5	ণ্ষ	0.4	জ্র	0.3	শ্ড	0.3	ল্ট	0.2	গ্ধ	0.2	ম্স	0.1
ক্ত্র	0.8	ণ্ড্য	0.5	স্ছ	0.4	ল্ড	0.3	ন্থ	0.3	দ্র	0.2	ল্শ	0.2	ণ্ট	0.1
ট্র	0.7	ণ্ড্র	0.5	হ্ম	0.4	শ্র	0.3	ত্ল	0.3	ট্র	0.2	য়্র	0.2	ল্ফ	0.1
ঙ্ক্য	0.7	ভ্র	0.5	ম্ক	0.4	শ্ভ	0.3	হ্ল	0.3	ম্ন	0.2	ণ্য	0.2	প্ফ	0.1
ক্ষ্ম	0.7	র্ব	0.5	ন্ব	0.4	স্র	0.3	ফ৵	0.3	ন্ঢ	0.2	ণ্ম	0.2	র্থ	0.1
ঙ্ক	0.7	ষ্ঠ	0.5	স্জ	0.4	ণ্ক	0.3	থ্ব	0.3	শ্ম	0.2	স্ট	0.2	ণ্ল	0.1
স্থ	0.7	শ্ছ	0.5	ট্র	0.4	ন্ব	0.3	ষ্ম	0.3	দ্র	0.2	প্ঢ	0.2	ল্স	0.1
ন্ত	0.6	দ্ম	0.5	দ্গ	0.4	ণ্গ	0.3	শ্ক	0.3	প্ট	0.2	খ্র	0.2	প্র	0.1
চ্ছ	0.6	দ্র	0.5	ড্ড	0.4	ন্য	0.3	স্ঠ	0.3	প্ঠ	0.2	র্ধ	0.2	ল্ল	0.1
ঙ্ক	0.6	খ্র	0.4	ন্ত	0.4	ড্গ	0.3	স্প	0.3	ন্ন	0.2	ত্র	0.2	স্গ	0.1
ঙ্ক্র	0.6	ক্ক	0.4	খ্য	0.3	প্ণ	0.3	গ্শ	0.3	ড্র	0.2	গ্শ	0.2	র্ফ	0.1
ষ্ণ	0.6	ক্ট	0.4	ক্ত	0.3	ঠ্র	0.3	প্ধ	0.3	ক্ষ	0.2	শ্ট	0.2	খ্য	0.1
প্ঙ	0.6	ম্ছ	0.4	ক্ষ	0.3	শ্হ	0.3	স্শ	0.3	ল্ন	0.2	শ্ঠ	0.2	র্দ	0.1
জ্ঞ	0.6	ম্ট	0.4	ম্চ	0.3	ণ্ট	0.3	দ্ধ	0.3	প্দ	0.2	ষ্ণ	0.2	র্ন	0.1
চ৵	0.6	ল্চ	0.4	প্গ	0.3	ণ্ঠ	0.3	ব্র	0.3	ড্ড	0.2	স্ড	0.2	র্স	0.1
স্ঙ	0.6	জ্ঝ	0.4	ঙ্ক	0.3	স্ক	0.3	ফ্ল	0.3	ল্দ	0.2	ন্র	0.2	র্প	0.1
ঙ্ক্ত	0.6	প্জ	0.4	স্স	0.3	শু	0.3	হ্ম	0.3	গু	0.2	প্হ	0.2	–	–
ষ্ট	0.6	ষ্ম	0.4	গ্ধ	0.3	শ্স	0.3	ব্দ	0.3	ব্ল	0.2	ষ্ট	0.2	–	–
ণ্জ	0.6	স্প	0.4	ক্র	0.3	ভ৵	0.3	প্ভ	0.3	ণ্ঠ	0.2	র্ব	0.2	–	–
ঙ্ক্য	0.6	জ্ত	0.4	গ্ন	0.3	ল্ম	0.3	ন্ম	0.3	ল্প	0.2	শ্জ	0.2	–	–
ঙ্ষ	0.6	জ্ম	0.4	ল্খ	0.3	ণ্ত	0.3	ক্ল	0.2	ণ্চ	0.2	দ্ব	0.2	–	–
ব্র	0.6	ছ্ম	0.4	গ্ন	0.3	ণ্ছ	0.3	খ্ব	0.2	ম্শ	0.2	শ্চ	0.2	–	–
হ্ম	0.6	ল্ঠ	0.4	খ্ভ	0.3	প৵	0.3	শ্থ	0.2	ণ্দ	0.2	ট্র	0.2	–	–
ষ্ভ	0.6	ঙ্গ	0.4	গ্র	0.3	ণ্ঙ	0.3	শ্ণ	0.2	স্ণ	0.2	স্স	0.2	–	–
ম্ঙ	0.5	ড়্গ	0.4	ক্ল	0.3	ল্ভ	0.3	গ্ব	0.2	প্ন	0.2	ন্ট	0.2	–	–
ঙ্ক	0.5	ষ্ম	0.4	ল্ল	0.3	ণ্ধ	0.3	গ্ম	0.2	ম্হ	0.2	হ্ম	0.2	–	–
ক্ষ	0.5	ন্ত	0.4	ম্জ	0.3	গ্ন	0.3	প্ক	0.2	ণ্ঢ	0.2	শ্ঢ	0.1	–	–
ল্ঙ	0.5	ঠ৵	0.4	ম্ঢ	0.3	ঙ্ঘ	0.3	প্খ	0.2	স্ন	0.2	ঙ্ম	0.1	–	–
ল্ঝ	0.5	ঙ্ভ	0.4	গ্ন	0.3	স্থ	0.3	শ্দ	0.2	প্ল	0.2	গ্য	0.1	–	–

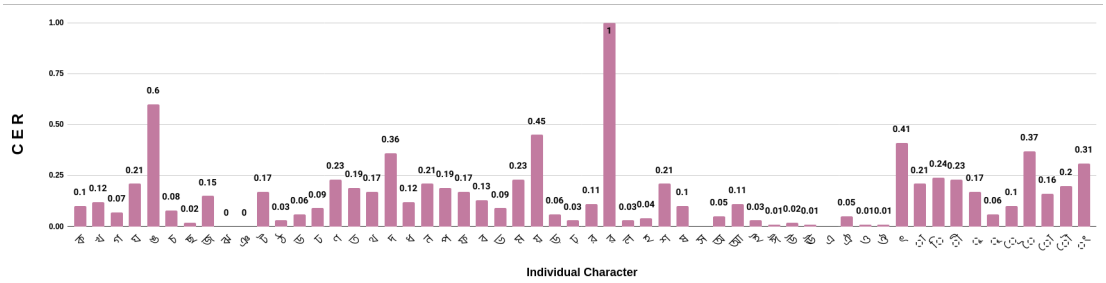


Figure 6.2: Character-level Error Rate (CER)[ranging from 0 to 1] of 56 individual characters provided by the Bengali OCR system on Manipuri texts.

6.3.2 Evaluation of the Bengali OCR system on Manipuri documents

As in the previous case, to evaluate the Bengali OCR system on Manipuri documents, this study considers the one provided by the Tesseract engine. Figure 6.2 and Table 6.4 show the CER of this OCR system for the individual and commonly used conjunct characters in Manipuri documents, respectively. Figure 6.2 shows that the character ৱ(Wa) has the highest CER with the value 1. It is followed by the character ঙ with the CER of 0.6. The average error rate of this OCR system for the individual character is 0.19 against 0.2 for the Assamese OCR system.

Like in the Assamese OCR system, the followings are noted.

- The character ৱ (Wa) is always recognized as one of the characters from ব (Ba) and র (ra) . Table 6.2 shows that the character ৱ in the word ৱারী is recognized as র , and as ব in the word ৰ্ৱক. Though the character ৱ is a legitimate character in Bengali script, it is not commonly used in Bengali documents. As a result, the underlying OCR model has not seen it while training.

- Like in the case of the Assamese OCR system, characters with similar structures are sometimes recognized interchangeably. Some of the examples are shown in Table 6.2.

Because of the character ৱ, the combined characters ৰ্ৱand ৱু have the highest CER

of 1. Out of the two combined characters formed with র i.e রু and র্, the Bengali OCR system fails to recognize র্ and hence it also has CER of 1. Table 6.4 shows the observed CER while recognizing conjunct characters. It has an average error rate of 0.2 against 0.31 of the Assamese OCR system. Like in the case of the Assamese OCR system, most of the errors in conjunct characters are because of their complicated and similar structures. The Bengali OCR system provides the CLA of 74.61% against 66.56% of the Assamese OCR system. Out of 50,499 characters in the MTD dataset, 12,734 characters were misclassified.

From the above analysis, it can be concluded that the performance of both Bengali and Assamese OCR systems on the Manipuri documents is poor. However, there is hope that they could be the base OCR models of the adaptation for the Manipuri OCR system. The next section presents the proposed adaptation strategies.

6.4 Adaptation strategies

As mentioned above, the objective of this chapter is to determine a suitable approach to adapt existing OCR systems for Manipuri documents with an acceptable performance. This section briefly discusses three adaptation strategies. All the proposed adaptation strategies considered retraining of baseline OCR system (Bengali or Assamese OCR) under different circumstances.

6.4.1 Random and linear increment strategy

It is understood from various studies that the OCR systems (with acceptable performance) that are developed using deep networks like LSTM may need a large volume of training samples. This strategy populates the training corpus linearly with randomly selected new samples and is referred to as *random and linear increment*. The motivation behind this setup is to estimate the approximate size of the training corpus, which is

Table 6.4: Character-level Error Rate (CER) provided by the **Bengali OCR system** for the 307 commonly used Manipuri conjunct characters. Column *Ci* presents conjunct characters and their respective CER.

C1		C2		C3		C4		C5		C6		C7		C8	
স্ঙ	0.9	ক্ল	0.5	ণ্ড্র	0.4	প্ল	0.3	চ্ব	0.2	ন্ম	0.1	ম্ণ	0.1	র্ফ	0.1
জ্ম	0.9	গ্ধ	0.5	ণ্প	0.4	ম্হ	0.3	চ্ব	0.2	ল্ল	0.1	প্চ	0.1	র্দ	0.1
ঙ্ব	0.8	গ্ল	0.5	ল্ষ	0.4	ণ্ড	0.3	চ্ছ	0.2	ল্স	0.1	ল্ণ	0.1	শ্গ	0.1
প্ঙ	0.8	ন্ত	0.5	ঝ্ম	0.4	প্ল	0.3	ক্র	0.2	ণ্ভ	0.1	ম্ধ	0.1	র্ন	0.1
ঙ্র	0.8	ম্ট	0.5	ঙ্ষ	0.4	র্থ	0.3	ল্জ	0.2	জ্র	0.1	স্দ	0.1	হ্ল	0.1
ব্ধ	0.8	ল্চ	0.5	স্প	0.4	ণ্শ	0.3	ঝ্ম	0.2	ভ্ল	0.1	চ্র	0.1	র্ধ	0.1
ঞ্চ	0.8	প্জ	0.5	হ্ন	0.4	ণ্ফ	0.3	ট্র	0.2	ত্ন	0.1	শ্ম	0.1	ত্র	0.1
জ্ঝ	0.8	চ্ব	0.5	ষ্ভ	0.4	প্শ	0.3	প্ছ	0.2	ল্শ	0.1	শ্ভ	0.1	শ্খ	0.1
ল্ঞ	0.7	ল্ত	0.5	ক্ষ	0.3	ত্ব	0.3	ট্য	0.2	স্ছ	0.1	ল্ন	0.1	ম্ক	0.1
চ্ছ	0.7	স্ক	0.5	খ্র	0.3	ণ্ম	0.3	প্ড	0.2	ম্ল	0.1	ণ্ক	0.1	স্ম	0.1
ন্ট	0.7	ণ্ড	0.5	ব্ধ	0.3	প্র	0.3	ল্ট	0.2	প্ল	0.1	প্দ	0.1	ষ্ম	0.1
ঙ্ক	0.7	ণ্ড্য	0.5	ক্ট	0.3	শ্ট	0.3	দ্র	0.2	খ্র	0.1	প্ণ	0.1	র্প	0.1
প্ভ	0.7	ভ্র	0.5	ম্ছ	0.3	শ্ঠ	0.3	জ্ণ	0.2	ল্হ	0.1	ল্দ	0.1	ন্য	0.1
ছ্র	0.7	গ্র	0.5	ক্ল	0.3	ন্ব	0.3	ন্ড	0.2	ম্ট	0.1	ণ্ত	0.1	স্ড	0.1
প্ঞ	0.7	শ্প	0.5	শ্ণ	0.3	র্র	0.3	ঙ্গ	0.2	শ্ড	0.1	শ্শ	0.1	ষ্ট	0.1
জ্ঝ	0.7	হ্ব	0.5	প্খ	0.3	ট্ব	0.3	শ্র	0.2	ন্থ	0.1	ভ্ব	0.1	ষ্ঠ	0.1
ষ্ট	0.7	ক্র	0.4	শ্চ	0.3	দ্গ	0.3	ড়্গ	0.2	ত্ল	0.1	ম্ল	0.1	র্ব	0.1
ষ্ক্য	0.7	গ্ন	0.4	গ্ম	0.3	শ্ছ	0.3	প্ঠ	0.2	ফ্ব	0.1	ল্ল	0.1	র্শ	0.1
ঙ্ক	0.6	ল্খ	0.4	গ্ল	0.3	ড্ড	0.3	ঢ্য	0.2	থ্ব	0.1	ণ্চ	0.1	র্ল	0.1
ল্ঝ	0.6	ল্ল	0.4	ঐ	0.3	ন্ত	0.3	ঙ্ঘ	0.2	স্হ	0.1	ল্ভ	0.1	হ্ম	0.1
শ্ব	0.6	গ্ল	0.4	স্ধ	0.3	ট্ট	0.3	ক্ষ্ণ	0.2	শ্ক	0.1	ণ্ধ	0.1	র্ম	0.1
শ্ধ	0.6	ল্ছ	0.4	ন্থ	0.3	দ্র	0.3	ন্ব	0.2	স্ঠ	0.1	স্ণ	0.1	র্ভ	0.1
ম্প	0.6	ঘ্ন	0.4	জ্ত	0.3	ল্ঙ	0.2	ণ্গ	0.2	ন্ন	0.1	ণ্ণ	0.1	র্স	0.1
ধ্র	0.6	শ্ফ	0.4	জ্ব	0.3	খ্য	0.2	ণ্খ	0.2	প্হ	0.1	স্ন	0.1	র্ষ	0.1
ড্গ	0.6	ষ্ণ	0.4	ছ্ব	0.3	ক্ত	0.2	ষ্ম	0.2	প্ধ	0.1	স্থ	0.1	দ্ব	0.1
ঐ	0.6	ল্ট	0.4	ট্ট	0.3	প্গ	0.2	শ্হ	0.2	দ্ধ	0.1	প্ফ	0.1	শ্চ	0.1
ট্ব	0.6	ঘ্ম	0.4	ল্ঠ	0.3	ক্ক	0.2	ণ্ট	0.2	স্শ	0.1	স্চ	0.1	র্য	0.1
ল্থ	0.6	ঞ্জ	0.4	ভ্র	0.3	স্স	0.2	ষ্ঠ	0.2	শ্জ	0.1	স্ত	0.1	–	–
শ্ষ	0.6	ষ্ম	0.4	ড্র	0.3	খ্ভ	0.2	প্থ	0.2	ব্র	0.1	ণ্ম	0.1	–	–
ত্ত	0.6	ন্ঢ	0.4	ণ্ট	0.3	গ্র	0.2	ব্ল	0.2	স্স	0.1	য্ব	0.1	–	–
ঙ্ক্য	0.6	ম্ন	0.4	ষ্ণ	0.3	ক্র	0.2	ন্দ	0.2	হ্ল	0.1	ণ্ন	0.1	–	–
ক্ত্র	0.6	ঠ্ব	0.4	ন্ন	0.3	ম্জ	0.2	ল্ম	0.2	ব্দ	0.1	ল্ল	0.1	–	–
ষ্ঠ	0.6	ট্র	0.4	স্গ	0.3	খ্ব	0.2	ণ্ত	0.2	শ্স	0.1	ণ্স	0.1	–	–
দ্ম	0.6	ড্ড	0.4	ঙ্ঘ	0.3	শ্থ	0.2	ম্স	0.2	ম্ঙ	0.1	প্ম	0.1	–	–
ফ্ল	0.6	ড্গ	0.4	ন্ধ	0.3	গ্ধ	0.2	প্ব	0.2	ম্চ	0.1	গ্ধ	0.1	–	–
ক্ষ্ম	0.6	ঙ্ক্ত	0.4	ঠ্র	0.3	ন্স	0.2	ণ্ঙ	0.2	গ্ম	0.1	য়্ম	0.1	–	–
ব্ধ	0.6	ণ্ড	0.4	ঢ্ব	0.3	ন্ত	0.2	ণ্দ	0.2	প্ক	0.1	ণ্ষ	0.1	–	–
হ্ল	0.6	ণ্ছ	0.4	স্জ	0.3	ম্চ	0.2	ণ্ট	0.2	শ্দ	0.1	ণ্য	0.1	–	–
স্থ	0.6	ণ্জ	0.4	ম্শ	0.3	চ্য	0.2	স্চ	0.2	গ্য	0.1	স্ট	0.1	–	–
ম্ঞ	0.5	ল্ব	0.4	ল্ফ	0.3	ম্ড	0.2	ণ্ঢ	0.2	ঘ্য	0.1	প্চ	0.1	–	–

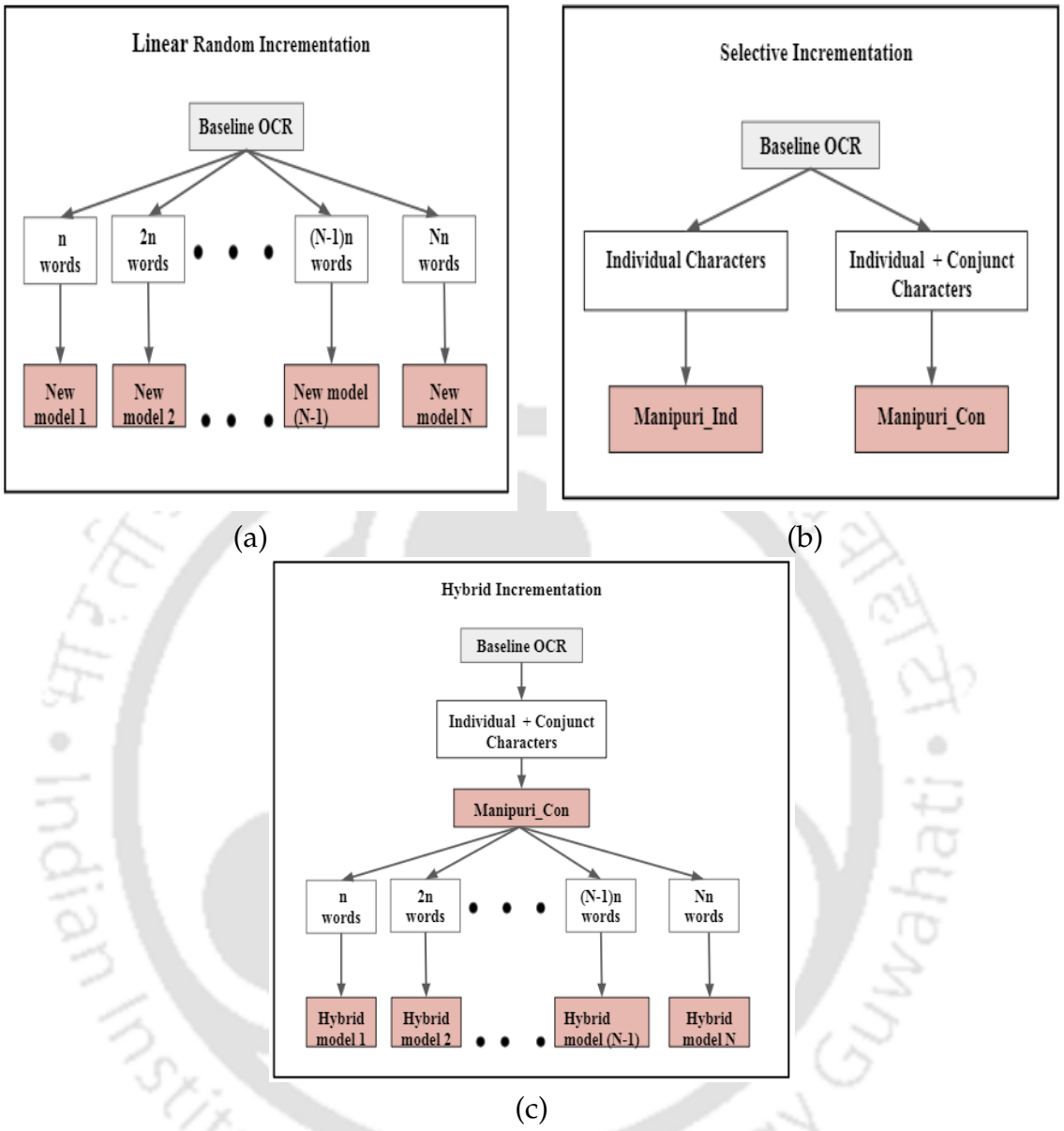


Figure 6.3: Pictorial representation of different adaptation strategies : (a) Random and linear increment, (b) Selective increment, and (c) Hybrid selective and linear increment.

likely to provide acceptable performance for Manipuri texts. Figure 6.3 (a) shows a pictorial presentation of this strategy. As shown in the figure, the training corpus is populated with additional n Manipuri words in each iteration. So, with this approach,

his study obtained n OCR models for a given baseline OCR model.

6.4.2 Selective increment strategy

In the random and linear increment approach, the training corpus is populated without any policy. However, the underlying OCR system has its own strength and weakness. The proposed *selective increment* evaluates the strengths and weaknesses of the base OCR model on Manipuri documents and improves upon the weaknesses. The motivation behind this approach is to enhance the performance of the OCR model by adding a few selective training samples. In Section 6.3, we have studied the error patterns in identifying the individual characters and conjunct characters. Based on the CER observed in this analysis, the selective adaptation identifies the list of individual and conjunct characters with high CER. The selected characters are then added to the training corpus. Figure 6.3 (b) shows the pictorial representation of the proposed selective increment strategy. As shown in the figure, it first investigates the performance of the OCR system by adding selected individual characters to the training corpus and the resultant model is denoted by Manipuri_Ind. It then adds the combination of selected individual and conjunct characters to the training corpus and the resultant model is denoted by Manipuri_Con. Hence, two OCR models are obtained for a given baseline OCR model.

6.4.3 Hybrid selective and linear increment strategy

As shown in Figure 6.3 (c), we apply both the above approaches. We first apply selective increment to build Manipur_Con OCR model, and then apply random and linear increment. The idea behind this approach is that selective increment will address major issues reflected out of the differences in the distribution of character usage in the two languages. It is then supplemented with random and linear increments to normalize

the presence of possible under-sampled characters. So, with this adaptation approach, we obtained n OCR models (n being the number of iterations in the random and linear increment approach) for a given baseline OCR model.

6.5 Performance of different adaptation strategies on the Assamese and the Bengali OCR system

As mentioned in the previous section, we consider the Assamese and the Bengali OCR systems provided by the Tesseract OCR engine $*$. This study regarded these two OCR systems as our baseline OCR models.

Tesseract is an OCR engine that offers robust support for Unicode (UTF-8) encoding and is capable of recognizing over 100 languages without requiring additional configuration. It provides several advantageous features. Firstly, Tesseract can be trained to recognize languages other than the default ones. This flexibility allows users to extend its language support based on their specific requirements. Secondly, it supports multiple output formats, including plain text, HTML, PDF, invisible-text-only PDF, and TSV, offering versatility in the output presentation. Additionally, the experimental support for ALTO (XML) output in the master branch adds further options for output customization. Starting from version 4.00, Tesseract introduced a new neural network-based recognition engine, significantly improving the accuracy of text recognition in document images compared to earlier versions. To achieve optimal performance, it is beneficial to have a larger training dataset and include more pages that closely resemble the documents Tesseract will be processing. While the existing model data provided for Latin languages covers approximately 400,000 text lines across 4,500 fonts, the availability of fonts for other scripts is limited. However, it is still necessary to train the neural

$*$We considered Tesseract based Assamese and Bengali OCR systems because they are publicly available and popularly used.

nets on a comparable number of text lines specific to those scripts. It's important to note that training the models can take several days to weeks, as opposed to just a few minutes to hours, due to the complexity and computational requirements of the process.

In our approach, we focus on adapting the existing OCR model by utilizing the fine-tuning method offered by Tesseract. Fine-tuning allows us to train the model on new data without making any changes to the underlying network architecture. This approach is particularly useful when dealing with data that is similar to the existing training data but has subtle differences, such as unique fonts or variations in symbol representation. Tesseract 4.xx (specifically version 4.1.0-rc4 in our study) provides two main types of fine-tuning: fine-tuning for specific fonts and fine-tuning for additional symbols. To prepare the training data, we modify the original text used in the training process. Fortunately, Tesseract offers training text for both Assamese OCR models and Bengali OCR models, which can be easily extracted and utilized to create new training data for fine-tuning with our specific text. To perform the fine-tuning, we rely on a set of tools and dependencies. These include Tesseract 4.1.0-rc4, GPL Ghostscript 9.26, a compiler for C and C++ (such as GCC or Clang), GNU Autotools (autoconf, automake, libtool, pkg-config), and libraries such as Leptonica, libpng, libjpeg, and libtif. These dependencies ensure that we have the necessary tools and resources to successfully carry out the fine-tuning process. For detailed information and step-by-step instructions on the fine-tuning process, please refer to Appendix B.2, where we present the training process provided by Tesseract.

6.5.1 Data preparation

This study used four main text corpora. We already mentioned about MTD, and this section presents the detail of the remaining three. All the texts in the corpora are

Table 6.5: Character-level Error Rate(CER) of some individual characters produced by the Assamese and the Bengali OCR system on Manipuri texts.

Produced by Bengali OCR system		**Produced by Assamese OCR system**	
Character	**CER**	**Character**	**CER**
ৱ	1	র	1
ঙ	0.76	ধ	0.89
য়	0.75	ৱ	0.82
ৎ	0.69	ব	0.82
দ	0.66	য়	0.82
ল	0.66	ণ	0.77
ল	0.65	ণ	0.74
ণ	0.65	◌ং	0.74

extracted from the Manipuri daily Naharolgi Thoudang∗.As mentioned above, it is the only Manipuri news publication in UTF-8. It allows us to generate the experimental textual corpora without manual transcription. The detail of these three corpora are given below:

1. **Manipuri Random Words (MRW):** It includes 61,000 words randomly selected from 112 news articles. The words consist of 18,6105 individuals and 57,934 conjunct characters.

2. **Individual Characters Oriented Manipuri Words (ICOMW):** It includes two separate sets of Manipuri words selected on the basis of the CER in recognizing the individual characters by the baseline OCR models. Some of the characters with their CER are shown in Table 6.5. Here 1 represents 100% misclassification while 0 represents correct classification. This study sets a threshold at 0.4, i.e., words containing a character with CER ≥ 0.4 are selected (we have experimented with various thresholds such as 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6, and found 0.4 to be the optimal threshold). So the set created for Bengali OCR consists of words

∗http://www.naharolgithoudang.in/web/

that have characters like ৱ , and ঙ and that created for Assamese OCR has words with characters like ৱ , and ধ. The set created for the Bengali OCR system has about 2,500 words and the set created for the Assamese OCR system has about 3,000 words.

3. **Conjunct Characters Oriented Manipuri Words (CCOMW):** Like ICOMW, it also consists of two sets with about 2,000 words each. One is generated based on the error analysis of the Bengali OCR system, and the other is based on the error analysis of the Assamese OCR system. The words are selected in the same manner as in ICOMW, and the only difference is that the focus is on the conjunct characters with high CER instead of individual characters. Some of the conjunct characters with their corresponding CER are shown in Table 6.3 and 6.4 for the Manipuri text provided by the Assamese and the Bengali OCR system, respectively. We set an error rate threshold of CER $\geq$ 0.3 (same as in the creation of ICOW, we have experimented with various threshold values) to select the words. The set created for the Bengali OCR system consists of words that have characters like স্ঙ , and প্ঙ , and that for the Assamese OCR system has words with characters like ষ্ঠ, and জ্ঝ .

Among these four datasets, MTD is used for evaluating the performance of the models obtained with the proposed three adaptation strategies and the proposed semi-supervised training approach (discussed in Section 6.6). MRW is the training dataset in which the texts or words are collected randomly and the other two, ICOMW and CCOMW, are created with selective words. All the datasets, both in the image and their respective corresponding text format, will be made available for public use.

Table 6.6: Character Level Accuracy (CLA) of two baselines OCR models on Manipuri texts with first adaptation strategy (random and linear increment strategy). The number under () denotes the increment in CLA from their respective Baseline.

Baseline model	**Original training corpus** *plus*			
	0 word	**20000 words**	**40000 words**	**60000 words**
Assamese	66.56	68.18 (+1.62)	70.62 (+4.06)	72.15 (+5.59)
Bengali	74.61	77.23 (+2.62)	80.09 (+5.48)	83.75 (+9.14)

6.5.2 Performance of the random and linear increment strategy

As discussed in Section 6.4, the training corpus of the baseline model is populated linearly with a random set of Manipuri words. The MRW corpus has been considered for this analysis. This corpus is divided into batches of about 20,000 words each, and the training corpus of the base model is progressively incremented with one batch, two batches, and three batches. Table 6.6 shows the performance of the baseline OCR models after the random and linear progressive increments. The first column of *Original training corpus plus* i.e., "0 word" presents the CLA of the original Assamese and Bengali OCR system on Manipuri text. An average improvement of about 2.5% and 3.5% per the increment of 20,000 random words is observed for Assamese and Bengali OCR models, respectively. After the addition of about 60,000 words, the Assamese and Bengali OCR systems are able to achieve the CLA of about 72.15% (improvement of 5.59%), and 83.75% (improvement of 9.14%), respectively. From the above observations, it is evident that the performance of the baseline models improves with the addition of new samples. However, the number of samples needed for higher accuracy may be very large. Considering the linear increment, we may need about 2,50,000 random words for the Assamese OCR system and about 1,50,000 random words for the Bengali OCR system to achieve a CLA of about 95% on Manipuri texts.

Table 6.7: Character Level Accuracy (CLA) of two baselines OCR models on Manipuri texts with second adaptation strategy (selective increment strategy). Number in red denotes the increment in CLA from their respective Baseline.

Baseline Model	**Vanilla**	**Manipuri_Ind**	**Manipuri_Con**
Assamese	66.56	72.97 (+6.41)	77.89 (+11.13)
Bengali	74.61	83.95 (+9.34)	89.91 (+15.3)

6.5.3 Performance of the selective increment strategy

As discussed in Section 6.4, the motivation behind this strategy is to enhance the performance of the baseline OCR model by adding selective new samples. The performance of the baseline OCR models is studied after retraining with their respective ICOMW datasets and then with the combination of their respective ICOMW and CCOMW datasets. Table 6.7 shows the CLA of the baseline OCR models (Vanilla), after retraining with their respective ICOMW datasets (newly obtained model is denoted by Manipuri_Ind), and the combination of their respective ICOMW and CCOMW datasets (newly obtained model is denoted by Manipuri_Con). After retraining with corresponding ICOMW, the Assamese and Bengali based Manipuri_Ind OCR model is able to achieve the CLA of 72.97% (improvement of 6.41%) and 83.95% (improvement of 9.34%) respectively. It may be noted that just by adding about 2,000 carefully selected words, we are able to achieve the performance equivalent to that obtained by adding 60,000 random words. Finally, after retraining with corresponding ICOMW and CCOMW, the Assamese and Bengali based Manipuri_Con OCR model achieved the CLA of 77.89% (improvement of 11.13%), and 89.91% (improvement of 15.3%) respectively.

6.5.4 Performance of the hybrid selective and linear increment strategy

As discussed in Section 6.4, this strategy combines the selective increment and the random increment strategies. The baseline OCR model is trained with ICOMW and CCOMW to obtain Manipuri_Con, which takes care of recognition errors due to the usages of Manipuri characters unseen by and unpopular in the baseline OCR model.

As seen above, just by including the unseen and unpopular characters in the respective training corpora, the baseline Assamese OCR model improves from 66.56% to 77.89% and the baseline Bengali OCR model improves from 74.61% to 89.91%. Though it takes care of the unseen and unpopular characters in the baseline models, we would further need to introduce random Manipuri texts to capture the distribution of different characters in the Manipuri language. We, therefore, add the samples from the MRW dataset to the training corpora in batches of 20,000 each.

Table 6.8 shows the CLA of the baseline OCR models with our last adaptation approach i.e., hybrid selective and linear increment strategy. With the batches of 20000 words over MRW, we obtained three variants of Manipuri_Con OCR model for a given baseline model. As shown in the table, with the increase of training text randomly, the CLA of baseline models (both Assamese and Bengali based Manipuri_Con) increase. After the addition of all texts in MRW and the combination of their respective ICOMW and CCOMW datasets, the last variants are obtained. With these last variants, the Assamese-based Manipuri_Con and the Bengali-based Manipuri_Con achieve CLA of 84.89% and 95.82%, respectively. The improvement is 18% for the Assamese OCR system and 21% for the Bengali OCR system with an addition of just about 60,000 random words and 4,000 carefully selected words. The last variants of Assamese-based Manipuri_Con and the Bengali-based Manipuri_Con obtained in this section are referred to as Assamese-based and Bengali-based Hybrid OCR models, respectively, in the remaining part of this chapter.

Table 6.8: Performance of two Manipuri_Con OCR models: MCA (Assamese based Manipuri_Con), and MCB (Bengali based Manipuri_Con) on Manipuri texts with the third adaptation approach (hybrid selective and linear increment). Number in red denotes the increment in CLA from their respective Manipuri_Con OCR models.

Baseline Model	**Original training corpus** *plus*			
	0 word	**20000 words**	**40000 words**	**60000 words**
MCA	77.89	81.08 (+3.19)	83.01(+5.12)	84.89 (+7.00)
MCB	89.91	91.02 (+1.11)	93.67 (+3.76	95.82(+5.91)

Remarks*:* From the above experimental observations, it is evident that the Bengali and the Assamese OCR system can be effectively adapted using carefully selected strategies and error analysis to achieve reasonable performance. From the above observations, a few important points can be noted:

- Character recognition error analysis of the baseline OCR models is essential to identify the characters or words to be added to the training corpus. With the addition of just about 4,000 carefully selected words (ICOMW and CCOMW), the baseline OCR models can achieve improvements that are comparable to adding 60,000 random words (MRW).
- While selective increment is vital to take care of unseen and under-used characters, the random increment is also important to take care of the random distribution of different characters.

6.6 Semi-supervised Training

The semi-supervised approach has been famous for building an effective system in a situation where there are a limited amount of labeled data and a large amount of unlabelled data. This section presents our proposed semi-supervised algorithm. From the given random Manipuri document images (unlabelled data), the proposed algorithm

chooses various words with specific criteria to be added to the training text and retrain the model. In other words, it is able to automatically populate the training dataset with specific words (extracted from randomly collected manipuri document images). The specific criterion is the high confidence-level score of the model while recognizing the words. Tesseract provides the *word confidence* (*WC*) score for every word it recognizes. It is defined by

$$WC = \frac{\sum_{i=1}^{n} CC_i}{n} \tag{6.5}$$

where CC_i is the *character level confidence* of i^{th} character in the given word and n is the total number of characters in the word. It is the percentage of the similarity of the image and the "ideal" whose recognition confidence would be 100%. A better CC implies a better WC.

Algorithm 1 presents the workflow of the proposed semi-supervised approach. It consists of two main stages: Training and Analysis. They are carried out for n iterations. The Training stage has three main tasks: (i) *extraction of high-confidence words from the input images*, (ii) *the addition of those words to the training corpus*, and (iii) *retraining of the model with the newly obtained corpus*. This study sets a threshold of WC to 90%, and hence considered the recognition of the words with $WC \geq 90\%$, confident, and are added to the training corpus. The reason for considering 90% is that words recognized with a confidence level lower than 90% sometimes contain errors. With empirical evaluation, it is found that setting the threshold too high resulted in a trade-off between accuracy and quantity. While a higher threshold increases the accuracy of the OCR model, it reduces the number of words available for training the model. On the other hand, a lower threshold increases the quantity of training data, but it includes more erroneous words that could negatively impact accuracy. With 90%, it is able to provide a balance between accuracy and quantity. This algorithm produces a new OCR model at each

iteration from the baseline OCR model. The size of training datasets may be different for different iterations because the dataset at iteration i depends on the number of words that are confidently classified by the OCR model obtained at iteration $i-1$. To obtain the final OCR model from a base OCR, these three major tasks in the training stage are repeatedly performed. To terminate this repeated process, we consider the CER of the OCR model, and the process is described in the Analysis stage. In this stage, the performance of the model is observed through the CER, and the performance consistency during the training is also examined. If the difference in the CERs of the OCR models obtained in the iterations i and $i-1$ is below a threshold E_d (we consider it as 0.05), then the process is terminated. The proposed algorithm has experimented with the OCR models obtained with (i) hybrid selective and linear increment strategy, in Section 6.5.4 and (ii) selective increment strategy, in Section 6.5.3. Further, to evaluate the effectiveness of the proposed algorithm, we investigate its performance with the baseline Assamese and Bengali OCR systems.

For this framework, we prepare a dataset consisting of 30 news articles images from the source *Poknapham*, Manipur's local news daily (written in *Shree Man0* font). Approximately it has 15,000 words. From this dataset, the WC score of the Manipuri words recognized by the two baseline OCR system is low. Baseline Assamese and Bengali confidently extract only 583 and 654 words, respectively. However, the WC score for some common words (that exist in the English language) like লীদর, ওপোজিসন and small length words like অসি , খাং are high. In another case, as expected, the number of words extracted confidently by the four adapted Manipuri OCR systems is high as compared to the two baselines OCR systems. An interesting observation is that the model Bengali base Hybrid OCR model provides a high WC score of the word "ইম্ফাল " whereas it returns a low WC score of the word " ইম্ফাল," which is the same word ইম্ফাল with a punctuation mark comma(","). From the newly created dataset, Assamese-based

Algorithm 4 Semi-Supervised Training

// Initialization //
Word as w
Total number of words in an input document as N
Training Dataset as T
Word Confidence score of the word w *w.r.t model* A *as* $WC_A(w)$
Character-level Error Rate of the model A *as* $CER(A)$
Threshold set by the user for iteration as Max

// Training //
while *terminate* $\neq 1$ **do**
  if *iteration* $== 0$ **then**
    Import base model as M
  else
    Import model obtained in the previous
    iteration $i-1$ *as* M
  end if
  while $j \neq N$ **do**
    if $WC_M(w_j) \geq 90$ **then** ▷ Threshold set to 90% for balancing accuracy and quantity
      $T_i \leftarrow T_i + w$
      $j \leftarrow j + 1$
    end if
  end while
  $M_{new} \leftarrow Train(M, T_i)$
  $terminate = Analysis(M_{new}, M)$
end while
$Final\_Model \leftarrow M_{new}$

// Analysis //

procedure ANALYSIS($Model_{current}, Model_{previous}$)
  $Diff_{CER} = CER(Model_{current}) - CER(Model_{previous})$
  if $Diff_{CER} \leq E_d$ **OR** *iteration* $==$ *Max* **then**
    Return 1
  else
    Return 0
  end if
end procedure

Manipuri_Con, Bengali-based Manipuri_Con, Assamese based Hybrid, and Bengali based Hybrid were able to confidently extract 6014 words, 6192 words, 6109 words, and 7084 words, respectively. These extracted words are fed as the respective training dataset for the first iteration, as shown in Table 6.9 (denoted as SST).

Table 6.9 shows the performance of the baseline OCR models (Assamese and Bengali OCR system), and four adapted OCR models: Assamese based Manipuri_Con

OCR model, Bengali based Manipuri_Con OCR model, Assamese based Hybrid OCR model, and Bengali based Hybrid OCR model under our proposed semi-supervised training. All their performance are evaluated using the testing dataset MTD and reported the CLAs, as shown in the table. Both baseline OCR models, Assamese and Bengali, gave their consistent performance of 67% in 3^{rd}-4^{th} iterations and 74.59% in 4^{th}-5^{th} iterations, respectively. Retraining the baseline Assamese and Bengali OCR models with our semi-supervised algorithm have no impact. It is because the added words (in each iteration) fail to train the models to learn Manipuri words. In the case of Manipuri_Con OCR models, the number of words that are classified confidently increases with every iteration for both Assamese-based and Bengali based. As shown in the table, the Bengali-based Manipuri_Con OCR model and the Assamese-based Manipuri_Con OCR model increased their CLA up to 96.03% and 83.14% respectively. Similar characteristics are observed for both the Assamese and Bengali-based Hybrid OCR models. The Assamese-based Manipuri_Con and the Bengali-based Manipuri_Con OCR models improved their CLA with the proposed semi-supervised algorithm to 97.76% and 85.87%, respectively. It is evident from these experiments that the proposed semi-supervised framework is suitable for adapting the OCR system.

Remarks*:* From the above observations, it can be said that the semi-supervised framework helps in improving the performance of the OCR models. For under-resource languages, a semi-supervised framework looks promising because we can automatically populate random texts without transcription. Considering the performance of the two baselines, two Manipuri_Con and two Hybrid OCR models on Manipuri texts, the following points are noted:

- *Characteristics of baseline OCR model:* At their best performance, the baseline Assamese and the Bengali OCR system are able to extract only 587 and 664 words confidently from about 14,894 Manipuri words. As they were trained on their

Table 6.9: Character Level Accuracy (CLA) of the two baselines OCR models and four adapted OCR models on Manipuri texts under the semi-supervised training. SST denotes the training text size, which in turn is the number of confidently recognized words by the model obtained in the previous iteration.

OCR		Iteration									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Baseline Assamese	SST	583	585	585	**587**	–	–	–	–	–	–
	CLA	66.57	66.58	67.00	**67.00**	–	–	–	–	–	–
Baseline Bengali	SST	654	661	661	664	**664**	–	–	–	–	–
	CLA	74.54	74.56	74.57	74.59	**74.59**	–	–	–	–	–
Manipuri_Con (Assamese based)	SST	6014	6281	6707	6921	7172	7194	7268	7304	7503	**7672**
	CLA	78.32	78.91	79.31	80.52	80.96	81.43	81.87	82.02	82.71	**83.14**
Manipuri_Con (Bengali based)	SST	6192	6811	6928	7021	7421	7501	7572	7594	8012	**8147**
	CLA	91.02	91.67	91.89	92.57	93.42	93.82	94.31	94.74	95.41	**96.03**
Hybrid (Bengali based)	SST	7084	7461	7905	8316	8594	8611	8653	8801	8926	**8947**
	CLA	95.78	96.17	97.04	97.19	97.41	97.53	97.58	97.62	97.71	**97.76**
Hybrid (Assamese based)	SST	6109	6219	6417	6794	6806	6817	6891	6911	6952	**6998**
	CLA	85.04	85.11	85.32	85.61	85.72	85.73	85.82	85.87	85.92	**85.97**

respective texts, they failed to recognize most of the Manipuri words correctly, and if they did, they recognized them with a low WC. Hence, the added texts in the training corpus do not favor training them to perform well on Manipuri documents.

- *Characteristics of* Manipuri_Con *OCR model:* Both Assamese based and Bengali based Manipuri_Con OCR models increase their CLA throughout ten iterations. As stated earlier, the number of words that are recognized confidently increases with the iterations. In the ten iterations, at maximum Assamese based Manipuri_Con OCR model, and Bengali based Manipuri_Con OCR model able to extract 7672 and 8147 words confidently from 14,898 words. For both, the rate of increase in the number of words that are classified confidently is slow.

- *Characteristics of* Hybrid *OCR model:* Following the same manner of Manipuri_Con

OCR models, both Assamese based and Bengali based Hybrid OCR models also increase their CLA throughout ten iterations. Assamese-based Hybrid OCR model is able to extract only 6998 words confidently from 14898 words. However, the Bengali based Hybrid OCR model extracts 8947 words confidently at the 10^{th} iteration. As shown in Table 6.9, the rate of increase in the number of words that are classified confidently for the Bengali based Hybrid OCR model is comparatively high than that of the Assamese-based Hybrid OCR model. It is further observed that among all four adapted OCR models, the Assamese-based Hybrid OCR model provides the smallest number of words (6998 words) that are classified confidently.

- For both the baseline OCR models, the size of their training corpus used to obtain Hybrid OCR models is large compared to that of the training corpus used to obtain Manipuri_Con OCR models. However, the difference in the CLA performance of the models obtained from the Hybrid and Manipuri_Con OCR models under semi-supervised training for both baseline OCR models are not significantly large. Hence, the inclusion of a large volume of random words in the training corpus for training the Manipuri_Con OCR model to obtain Hybrid OCR models has a lower impact during the semi-supervised training.

- The advantage of error analysis can be seen from the performance of the baseline OCR models, and four adapted OCR models during the semi-supervised training. Performing the semi-supervised training on the baseline OCR models has no considerable effect. To some extent, the resultant OCR models remain the same as that of baseline OCR models. However, the adapted four OCR models, which we obtained after performing error analysis on the baseline OCR models, are able to improve their CLA on Manipuri documents and produce better results.

6.7 Performance of the adapted Manipuri OCR under different environments

From the four adapted Manipuri OCR models obtained with semi-supervised training, for both baseline OCR models, we considered the one obtained with the Hybrid OCR model for further investigation. We have two OCR models, one from each baseline. The one obtained from the baseline Assamese OCR model that achieved a CLA of 85.97% under semi-supervised training is termed as Assamese-based Manipuri OCR system. The other one obtained from the baseline Bengali OCR model that achieved a CLA of 97.76% under semi-supervised training is termed as Bengali-based Manipuri OCR system. As the Bengali-based Manipuri OCR system performs better than the Assamese-based Manipuri OCR system, we further investigate its performance on various Manipuri documents. We considered two different environments : (i) Homogeneous environment and (ii) heterogeneous environment. In the Homogeneous environment, the testing datasets are made from the same source as the training datasets, while in the latter case, the testing datasets are created from different sources.

Under both the homogeneous and heterogeneous environments, the testing datasets of the following three different natures are considered. They are as follows.

1. Computerised Print Dataset: The words are copied from the source as a text file. The necessary editing is done on the text file, which is saved as a PDF file. Finally, the PDF file is converted to a TIFF file with 300 dpi.

2. Scanned Image Dataset: The words from the source are printed on paper and scanned with 300 dpi.

3. Mobile Captured Dataset: The words from the source are printed on paper, and the images are captured with a mobile camera of 48MP.

The details about these environments and the performance of the Bengali-based

Manipuri OCR system under these environments are discussed below.

6.7.1 Homogeneous environment

As mentioned above, in this environment, the words for the testing datasets are collected from the same source as that of the training. The Naharolgi thoudang, a local Manipuri daily, is considered as the source of this environment. We prepared a testing dataset consisting of 17 pages with 7,580 words. We evaluated the performance of the Bengali-based Manipuri OCR system with the test samples collected under the above imaging setups with the following two different scenarios.

1. Uniform fonts: The fonts of the testing and training dataset are kept uniform. As mentioned earlier, we use only four fonts, namely, Bengali Medium, Lohit Bengali, Mukti Narrow and Shree Man07. Therefore this scenario has four testing datasets of different fonts with identical contents.

2. Non-uniform fonts: The fonts of the testing and training datasets are different. We consider two fonts Ami and Lukhjam. So this scenario has two testing datasets of different fonts with identical contents.

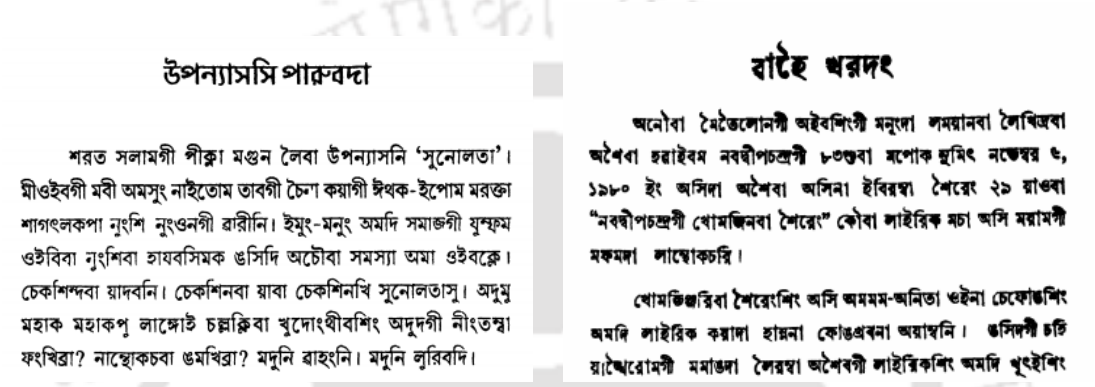
Table 6.10 shows the performance of the Bengali-based Manipuri OCR system under this environment. The following points are noted:

- Considering the uniform fonts scenario, the system achieves an average CLA performance of 97.22%, 97.16%, 96.11%, and 96.64% when the fonts are Shree Man07, Bengali, Mukti and Lohit respectively.

- In the second scenario, when the fonts are Ani and Lukhjan, regardless of the nature of the datasets, the system manages to attain an average CLA of 96.43% and 95.37% respectively.

Table 6.10: Character Level Accuracy (CLA) of the Bengali-based Manipuri OCR system under the homogeneous environment.

Type of Input Document	Uniform fonts				Non-uniform fonts	
	Shree Man07	Bengali	Mukti	Lohit	Ani	Lukhjan
Computerised Print	97.44	97.36	96.03	96.69	97.26	96.41
Scanned Image	97.18	97.01	96.12	97.04	95.93	96.27
Mobile Captured	97.04	97.11	96.18	96.21	96.20	93.43



(a) (b)

Figure 6.4: Input samples from Heterogeneous environment : (a) Khomjinba Lairik Khara, (b) Nabadi Chandragi Khomjinba Seireng.

In spite of differences in fonts, the Bengali-based Manipuri OCR system manages to attain an average CLA performance of 96.78% under the uniform fonts scenario and 95.91% under the non-uniform scenario.

6.7.2 Heterogeneous environment

Under this environment, the testing datasets are created from sources that are different from that of the training source. We chose four different sources (two local Manipuri dailies and two printed books). The details of each source are discussed below.

1. **News Papers:** We consider two local dailies namely, Poknapham∗ and Sangai Express†. About 7,720 words are extracted from 17 pages of each source. For each source, we generated six testing datasets (each consisting of 7,720 words) of different fonts with identical contents.

2. **Printed Books:** We have collected scanned images of 21 pages from Khomjinba Lairik Khara‡ and 25 pages from Nabadi Chandragi Khomjinba Seireng§. These books are available in the form of scanned images in JP2 format. The underlying fonts are unknown to the authors but they are very similar to one trained font, *Bengali Medium* . We have selected these two books for evaluation because of the differences in the image quality. Figure 6.4 shows a sample image taken from each of the books. The ground truth text for these printed documents is prepared manually.

Table 6.11 shows the performance of the Bengali-based Manipuri OCR system under this environment. The following points are noted.

- When the sources are Poknapham and Sangai Express, regardless of different fonts and nature of documents, the system gives a promising performance. It achieved an average CLA performance of 96.22% on the dataset created from the source Poknapham and 96.56% on the dataset created from the source Sangai Express.

- With the sources Khomjinba Lairik Khara and Nabadi Chandragi Komjinba Seireng in which the datasets are available only as scan documents, the Bengali-based Manipuri OCR system achieved a performance of 94.83% and 87.23% respectively. The performance is inferior for the latter case because of the low quality of the input images.

∗http://poknapham.in/
†https://thesangaiexpress.com/manipuri/
‡https://archive.org/details/in.ernet.dli.2015.466088
https://archive.org/details/in.ernet.dli.2015.465890

Table 6.11: Character Level Accuracy (CLA) of the Bengali - based Manipuri OCR system under heterogeneous environment.

Source	Font	Type of Image		
		Computerised Print Document	**Scanned Image Document**	**Mobile Captured Document**
Poknapham	Shree Man07	97.31	96.21	96.01
	Bengali	97.15	96.11	96.09
	Mukti	96.23	96.28	95.75
	Lohit	97.21	95.39	95.16
	Ani	96.84	96.17	96.14
	Lukhjan	96.87	95.09	96.00
Sangai Express	Shree Man07	97.39	96.74	96.37
	Bengali	97.27	96.67	97.08
	Mukti	96.85	96.72	96.98
	Lohit	97.38	96.81	96.42
	Ani	96.78	95.72	95.04
	Lukhjan	97.14	95.65	95.11
Khomjinba Lairik Khara	Unknwon	–	94.83	–
Nabadi Chandragi Khomjinba Seireng	Unknwon	–	87.23	–

Thus the Bengali-based Manipuri OCR system achieves a promising performance under both homogeneous and heterogeneous environments. However, its performance degrades on printed books when the image quality is below average.

6.7.3 Cross-lingual performance of adapted OCR systems

We investigate the performance of the two adapted Manipuri OCR systems (i) Assamese-based Manipuri OCR system (ii) Bengali-based Manipuri OCR system, obtained in Section 6.6, on the cross-lingual documents. The preparation of the testing documents for the Assamese and Bengali languages is described below.

Table 6.12: Cross-lingual performance (given in CLA) of the baseline Assamese, baseline Baangla and two adapted Manipuri OCR systems.

OCR System	Testing document		
	Bengali-Doc	**Assamese-Doc**	**Manipuri-Doc**
Assamese based Manipuri	77.27	81.92	84.64
Bengali based Manipuri	91.72	85.19	97.31
Assamese	79	94.03	63.58
Bengali	97	66.56	74

1. Bengali testing document: A text document of 17 pages consisting of about 7500 words is extracted from the Bengali news publication Prothom Alo∗. This document is generated using the *SolaimanLipi* font which is unseen by the adapted Manipuri OCR systems.

2. Assamese testing document: A text document of 17 pages consisting of about 7500 words is extracted from the Assamese news publication Asomiya Pratidin†. This document is generated using the *Kalpurush* font, which is unseen by the adapted Manipuri OCR systems.

In addition, we investigate the performance of the baseline Assamese and the baseline Bengali OCR system on a Manipuri document of about 7500 words generated in the font which is unseen by them. We consider the text extracted from the *Poknapham* news daily as discussed in the previous section and generate the document in Shree Man07 font.

Table 6.12 shows the cross-lingual performance of the OCR systems. It can be seen that the baseline Assamese and the baseline Bengali OCR systems are capable of extracting text with high accuracy only from the documents in their respective languages.

∗https://www.prothomalo.com/
†https https://www.asomiyapratidin.in/

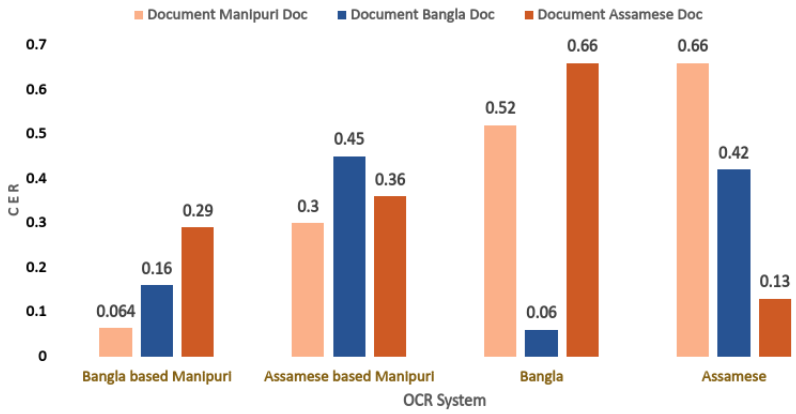


Figure 6.5: Changes in CER of baseline Bengali, baseline Assamese, Assamese-based Manipuri and Bengali-based Manipuri OCR system on cross-lingual documents.

Interestingly, adapted Manipuri OCR systems perform comparably on the documents of other languages. For example, the Assamese-based Manipuri OCR system gives the CLA of 81.92% on the Assamese documents compared to the CLA of 84.64% on the Manipuri documents. Similarly, the Bengali-based Manipuri OCR system gives a comparable CLA of 91.72% on Bengali documents compared to the CLA of 97.31% on Manipuri documents. These indicate that even after adapting to the Manipuri language, the adapted OCR systems maintain the original capability of the respective base languages. One interesting point is that the Bengali-based Manipuri OCR System performs better on Assamese documents than the Assamese-based Manipuri OCR system. Figure 6.5 shows the changes in the CER of these four OCR systems on cross-lingual documents.

On the basis of all these experiments and analysis, we conclude that the Bengali-based Manipuri OCR system can be adopted as a Manipuri OCR system.

6.8 Conclusion

To develop a Manipuri OCR system with limited available resources, this study proposed three different strategies to adapt the existing Assamese and Bengali OCR systems for

Manipuri documents. It established the importance of performing error analysis. In contrast to the OCR systems adapted by adding random Manipuri words, the systems adapted on the basis of error analysis have better character-level accuracy. The adapted OCR models give even superior performance in the extended experiment of semi-supervised training, in which the training text is further populated automatically with random words which can be recognized with high confidence.

Our experiments suggest that the Manipuri OCR system adapted from the baseline Bengali OCR system can be adopted as the Manipuri OCR system. Its performance varies with the type of document and is superior when the document type is a digital print. It is comparable with the baseline Bengali and the baseline Assamese OCR systems. In addition to this, unlike the Assamese and the Bengali OCR systems, which fail to work effectively in the Manipuri documents, it performs well on the Assamese and Bengali documents. This study did not consider input images in extremely bad conditions like degraded images, historical images with torn characters, and characters with extremely different fonts. Considering all these aspects will improve the robustness of this system, and this is a part of our ongoing research work.

6.9 Summary

This chapter discussed the proposed method of the "Manipuri OCR" module for the Document Conversion System (DCS). The module focuses on transforming the segmented textual regions into an editable format, making them ready for various textual operations. Figure 6.6 visualizes the status of DCS development after the completion of this module. As depicted in the figure, the three major tasks and contributions of this thesis, namely (i) region segmentation and classification, (ii) chart type classification, and (iii) Manipuri OCR, has been successfully accomplished. To provide a comprehensive view

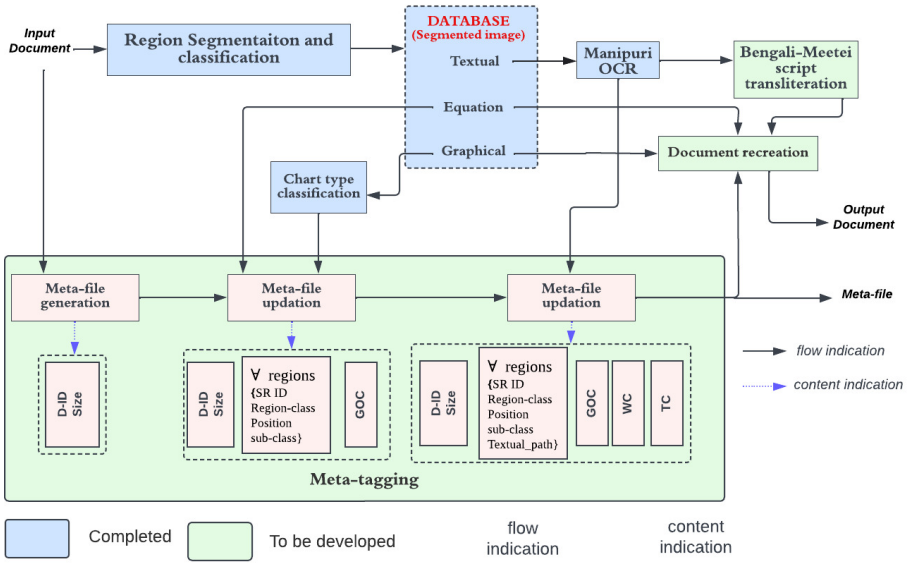


Figure 6.6: Status of the development for the proposed framework.

of the complete implementation of DCS, the integration of the remaining three modules is presented in Appendix A. It provides detailed information on the integration of the Bengali-Meetei script transliteration system, Meta-tagging, and Document recreation modules, resulting in the complete implementation of DCS.

7

Conclusion and Future work

This thesis addresses the urgent need for a Document Conversion System (DCS) specifically designed for Manipuri document images are written in the Bengali script to be converted into Meetei Mayek. The DCS aims to preserve the structural properties of the input document while producing two outputs: a document in the target script and a meta-file containing descriptive information. To implement the conceptualized DCS, six main modules are required: *region segmentation and classification, chart type classification, Manipuri OCR, Bengali-Meetei script transliteration, meta-tagging*, and *document recreation*.

However, this thesis primarily focused on the first three modules. The research works carried out in these modules are approached from three different perspectives: identifying research gaps and challenges in existing methods, developing appropriate data corpora, and addressing the identified research gaps. The first module, document segmentation, and classification system, tackles the challenge of improper segmentation of graphical regions with sparsely clustered pixels. This thesis proposed a hybrid approach combining deep learning and rule-based methods. Additionally, a dataset of 4,840 document images along with their respective mask or ground truth images was developed for this module. The second module, the chart classification system, introduces a deep learning-based model that utilized attention mechanisms and a triplet loss function. It specifically addressed two major challenges: chart noise and confusing chart-pairs. The thesis presents the largest chart dataset to date, consisting of 28 chart types and 117,271 samples. Lastly, the third module, Manipuri OCR system, focuses on developing an OCR system for Manipuri text written in the Bengali script. It provides a definitive answer to the question of whether Manipuri Bengali-scripted documents can be processed using OCR systems designed for other languages with a similar script, such as Assamese OCR or Bangla OCR. Furthermore, a Manipuri text corpus comprising 72,634 words is contributed in this thesis. The complete implementation of the conceptualized DCS, including the integration of the remaining three modules developed using existing technologies, is presented in Appendix A of this thesis.

In the future, there are plans to further extend and enhance this thesis by incorporating deep learning-based techniques for the implementation of the Manipuri Bengali-Meetei script transliteration system. This extension aims to leverage the advancements in deep learning models and algorithms to improve the accuracy and efficiency of the transliteration system. Additionally, the thesis intends to enhance the developed model of document layout analysis to address the challenges and limitations discussed in Chap-

ter 3. These enhancements may involve refining the segmentation and classification algorithms, exploring alternative approaches, and incorporating advanced deep learning techniques to achieve more accurate and robust document layout analysis results. Furthermore, the chart classification models presented in this thesis can be expanded to include a wider range of chart types. By incorporating additional chart types and training the models on larger and more diverse datasets, the accuracy and versatility of the chart classification module can be improved, enabling it to handle a broader range of data visualization scenarios. Overall, the future direction of this thesis includes incorporating deep learning techniques into the transliteration system, enhancing the document layout analysis model, and expanding the chart classification capabilities. These efforts aim to advance the field of Manipuri document processing and contribute to the development of more sophisticated and accurate systems for handling Manipuri documents.

A

Document Conversion System

As mentioned in Chapter 1, DCS is based on the combination of multiple sub-modules namely *document segmentation and classification, chart type classification, Manipuri OCR , Bengali-Meetei script transliteration, meta-tagging,* and *document recreation*. Among them, three sub-modules, viz., document segmentation and classification (developed in Chapter 3) , chart type classification (discussed and developed in Chapters 4, and 5), and Manipuri OCR (developed in Chapter 6) were discussed and developed. The development status of DCS is shown in Figure A.1. The remaining three modules,

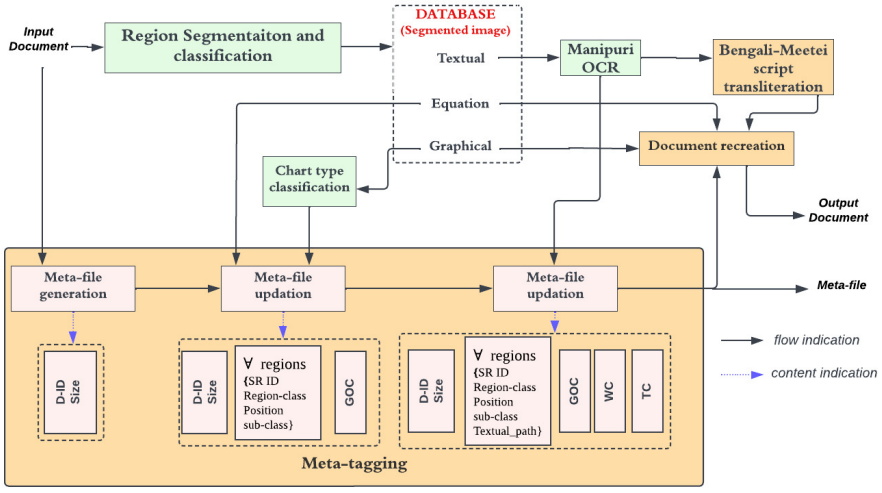


Figure A.1: Status of the development for the proposed framework.

namely, the *Bengali-Meetei script transliteration, meta-tagging,* and document recreation, are discussed in this Appendix. In addition, it presents the integration of all the modules to visualize the working of DCS.

The rest of this Appendix is organized as follows: In Section A.1, the development of the Bengali-Meetei script transliteration system is presented. Section A.2 talks about the meta-tagging process to create a meta-file. How we recreate the documents considering the online available tool is presented in Section A.3. Finally, Section A.4 presents the integration of all the modules to get DCS . It further discusses the evaluation of DCS against multiple datasets developed in different environments.

A.1 Bengali-Meetei script transliteration system

Transliteration refers to the process of converting words from one language script to another. In the context of Indian languages, transliteration has been extensively studied and applied in projects such as IT3, which was developed collaboratively by IISc Bangalore, India, and Carnegie Mellon University[43]. However, there is a limited number of machine transliteration studies specifically focused on Indian languages, as mentioned

in the study[75]. Some notable works in the field of Indian language transliteration include[132],[30],[36],[35], and[85].

For the Manipuri language, which has limited transliteration studies, works such as[96] and[126] are available. However, due to the lack of comparable text corpora for Manipuri in Bengali and Meetei scripts, this work adopts an existing rule-based approach proposed by[96] to develop a Bengali-Meetei script transliteration system.

A.1.1 Bengali script and Meetei script

The Bengali script consists of 52 consonants and 12 vowels. On the other hand, the Meetei script has 27 letters (Iyek Ipee shown in Table A.1), 6 independent vowel signs (shown in Table A.2), 8 dependent vowel signs(Cheitap Iyek shown in Table A.3), 8 final consonants (Lonsum Iyek shownin Table A.4), 10 digits (Cheising Iyek shown in Table A.5), and 3 punctuation marks (Cheikhei, Lum Iyek, and Apun Iyek). The transliterated text-line in Figure A.2 demonstrates the conversion between these two scripts.

In the Meitei script, certain alphabets are repeated to represent different Bengali alphabets. For example, the Bengali letters ষ, স , শ are transliterated as in the Meitei script.

Table A.1: Meitei script characters: Iyek Ipee.

Iyek Ipee			
ক->ꯀ (kok)	স(ছ,শ,ষ)->ꯁ (Sam)	ল->ꯂ (Lai)	ম->ꯃ (Mit)
প->ꯄ (Pa)	ন->ꯅ (Na)	চ->ꯆ (Chil)	ত(ট)->ꯇ (Til)
খ->ꯈ (Khou)	ঙ-> ꯉ (Ngou)	থ(ঠ)->ꯊ(Thou)	ব->ꯅ (Wai)
য(য়->ꯌ (Yang)	হ->ꯍ (Huk)	ঊ(ঊ)->ꯎ(Un)	ই(ঈ)->ꯏ(Ee)
ফ->ꯐ (Pham)	অ->ꯑ (Atia)	গ->ꯒ (Gok)	ঝ->ꯖ (Jham)
র->ꯔ (Rai)	ব->ꯕ (Ba)	জ-> ꯖ (Jil)	দ(ড)->ꯗ(Dil)
ঘ->ꯘ (Ghou)	ধ(ঢ)->ꯙ(Dhou)	ভ->ꯚ(Bham)	

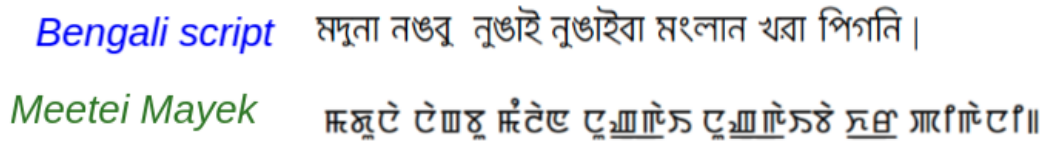


Figure A.2: Two scripts of Manipuri language/ Meeteilon: Bengali script (borrowed script), and Meetei script (original script).

Table A.2: Meitei script characters: Vowels.

Vowel letters			
আ->ꯑ(Aa)	এ->ꯑꯦ(Ae)	ঐ-ꯑꯤ(Ei)	
ও->ꯑꯣ(o)	ঔ->ꯑꯨ(Ou)	অং->ꯑꯪ(Ang)	

Table A.3: Meitei script characters: Cheitap Iyek.

Cheitap Iyek			
◌ো->ꯣ (ot nap)	ি, ী-> ◌ꯤ(inap)	া->◌ꯥ(aatap)	◌ে-> ◌ꯦ(yetnap)
◌ৌ-> ◌ꯧ (sounap)	ু, ূ->◌ꯨ (unap)	ৈ->◌ꯩ(cheinap)	◌ং-> ◌ꯪ(nung)

Table A.4: Meitei script characters: Cheising Iyek.

Lonsum Iyek			
ক্-> ꯀ (kok lonsum)	ল্-> ꯂ (lai lonsum)	ম্->ꯃ (mit lonsum)	প্-> ꯄ(pa lonsum)
ণ্, ন্-> ꯅ (na lonsum)	ট্,ত্-> ꯇ (til lonsum)	ঙ্->ꯉ(ngou lonsum)	ই, ঈ->ꯍ(ee lonsum)

Table A.5: Meitei script characters: Lonsum Iyek.

Cheising Iyek(Numeral figure)			
১->꯱(ama)	২->꯲(ani)	৩->꯳(ahum)	৪->꯴(mari)
৫->꯵(manga)	৬->꯶(taruk)	৭->꯷(taret)	৮->꯸(nipal)
৯->꯹(mapal)	১০->꯱꯰(tara)		

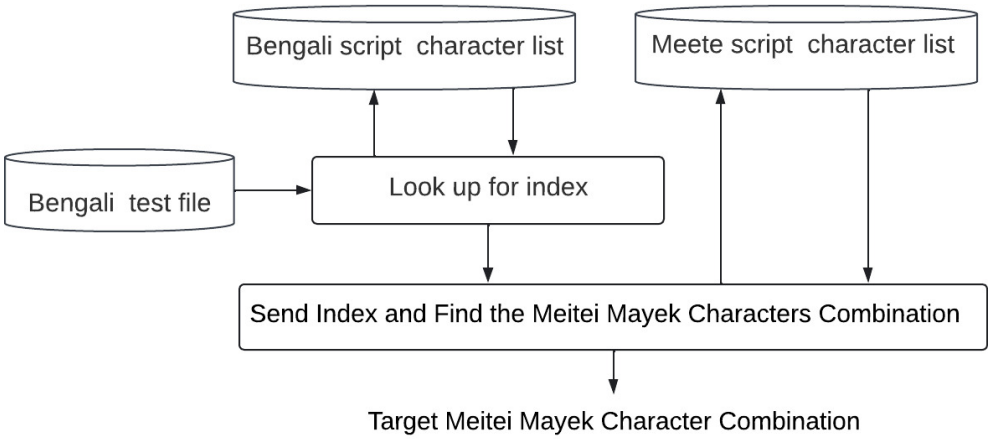


Figure A.3: Workflow of the Bengali-Meetei script transliteration system.

A.1.2 Rule-based transliteration system

The Bengali-Meetei script transliteration system is implemented based on the methodology presented in the study by Nongmeikapam[96]. The workflow of the system is illustrated in Figure A.3. The system involves the following steps:

- The Bengali characters and their corresponding Meetei script characters are mapped and stored in two separate lists, namely "Bengali" and "Meetei".
- A test file containing Bengali text is provided as input.
- The system reads the test file and for each character, it looks up its index in the Bengali Characters List.
- Using the index, the system retrieves the corresponding transliterated Meetei script character combination from the Meetei Characters List.
- The transliterated Meetei script character combination is stored in an output file.

The detailed process is described in Algorithm 5. This workflow enables the transliteration of Bengali text into the Meetei script, utilizing the predefined mapping between

Algorithm 5 Bengali-Meetei script transliteration

```
transliteration(line, TBC, Meetei[], Bengali[])
statement: Bengali line read from the document
TBC: Total number of Bengali Character
Bengali[]: The list of Bengali characters
Meetei[]: The list of Meitei script
Length: Length of the statement
for each m ≤ length do
    textline=statement.substring(m,m+1)
    if text-line equals blank space then
        print a white-space
    else

        for each index ≤ TBC − 1 do
            if textline equals Bengali[index] then
                position = index
                break
            end if
        end for
        Print the String Meetei[position] in the target file
    end if
end for
```

the two scripts. As Manipuri is a less digitized language, making a corpus is a difficult task. The corpus from a local Manipuri daily newspaper∗ is used for the trials with the systems. A corpus of 1000 words is compiled for system testing. The system has accrued a character-level-accuracy of 85.03% in the outcome evaluation.

A.2 Meta-tagging

Meta-tagging is the process of creating a meta-file. As defined in Chapter 1, its content can be grouped into three categories: (i) *document structures*: document size, and positional information of segments or regions (textual/non-textual) , (ii) *document contents (textual and non-textual)*: textual and non-textual contents in the document are saved as

∗https://naharolgithoudang.co.in/

the individual segmented images, and (iii) *content description*: descriptive information of the contents, such as number of words, number of text lines, number of graphical components, content types, image classes (like a chart, equation, and picture). As shown in Figure A.2, this process is carried out alongside the modules *Region segmentation and classification, Chart type classification, Manipuri OCR*, and *Bengali-Meetei script transliteration*. The meta-file is generated once DCS sees the input image. It records the image's size and it labels the input document. As the working of DCS proceeds, the content of a meta-file is updated, and finally it contains the wholesome information of the given input document image.

A.3 Document recreation

This module recreates the given input document image into the target scripted document with all the components at their respective positions. As shown in Figure A.1, this module expects three inputs: meta-file, equation and graphical segmented images from the DATABASE, and transliterated textual components, which are the outputs of the module *Bengali-Meetei script transliteration*. The recreation of the document is done using the open source tool *TeXStudio*∗. The recreation of the document is done in the following steps:

- *Creation of blank page:* A blank page with the specific size (width and height must match to the size of the given input document image, which can be retrieved from the meta-file) is created with a *TeXStudio* package **\geometry**. The unit is user defined. As we are considering the input document in image, we consider the pixel as unit which can be represented by *pt* in *TeXStudio*.

∗https://www.texstudio.org/

- *Insertion of textual components:* The Meetei scripted textual components, which are the outputs of transliteration system, is inserted to the blank page (created above) at their corresponding positions using two *TeXStudio* package **\tikz** and **\minipackage**. It is done in sucha a way that the textual components are first inserted in a minibox provided by **\minipackage**, and finally the box is inserted at the original textual position using **\tikz**. *TeXStudio* converts plain text files to UTF-8. However, Meetei script is UTF-16 code format, occupying U+ABC0 to U+ABFF in the Unicode block. In this case the question is about typesetting the Meetei script. This can be done using XeLetex, the fontspec package, and a suitable font (in this study, we are using Google Noto Sans Meetei script). To fit in a minibox, the fontsize of inserted text is set to small.

- *Insertion of equation and graphical components:* The components of equations and graphical regions are inserted into the blank page as images (retrieved from DATABASE). Their segmented images are put in their corresponding positions using a *TeXStudio* package **\picture**.

A.4 Document conversion

This section presents the DCS and its evaluation against multiple datasets. Re-visualizing the schematic workflow of DCS, it can be seen that it is the integration of the six modules which we have discussed in three chapters (Chapter 3, chapter 5, and chapter 6) and this Appendix. The performance of the DCS relies on the performance of its sub-modules particularly **document segmentaiton and classification**, **Manipuri OCR**, **Chart type classification** , and **Bengali-Meetei script transliteration**. To evaluate the performance of DCS, this thesis curated seven document image datasets which are discussed below:

- **Textaul Dataset (DS1):** This dataset consists of 300 document images with only text components. The texts are written in Manipuri Bengali script.
- **Equation Dataset (DS2):** This dataset consists of 200 document images with only equation components. The equation's components are the general mathematical expression written in English numerical, Greek, and Latin script.
- **Graphical Dataset (DS3):** This dataset consists of 100 document images with only graphical components. The graphical regions may have some embedded textual components written in Latin script.
- **Textual-Eqution Dataset (DS4):** This dataset consists of 500 document images with two components- textual and equations. It is made with the components in DS1 and DS2.
- **Textual-Graphical Dataset (DS5):** This dataset consists of 400 document images with two components- textual and graphical. It is made with the components in DS1 and DS3.
- **Equation-Graphical Dataset (DS6):** This dataset consists of 300 document images with two components- equation and graphical. It is made with the components in DS2 and DS3.
- **Textual-Equation-Graphical Dataset (DS7):** This dataset consists of 500 document images with all three components- textual, equation, and and graphical. This is made with the components in DS1, DS2 and DS3.

DCS is evaluated against the aforementioned seven datasets. Depending on the nature of dataset, the performance of the system is evaluated. For example, if the document image consists of only graphical and equation components, then the module *Manipuri OCR* and *Bengali-Meetei script transliteration* won't contribute to the evaluation. Let Θ

Table A.6: Performance of the proposed system, DCS (measured in accuray).

Dataset	Document segmentation		Chart type classification	OCR	Machine transliteration	DCS
DS1	Text	99.02	–	97.45	86.32	**94.26**
DS2	Equation	98.34	–	–	–	**98.34**
DS3	Graphical	95.67	98.06	–	–	**94 .50**
DS4	Text	92.56	–	96.32	85.24	**94.63**
	Equation	93.42	–	–	–	
DS5	Text	98.75	–	97.21	86.32	**95.32**
	Graphical	97.74	98.53	–	–	
DS6	Equation	98.53	–	–	–	**95.88**
	Graphical	94.93	98.18	–	–	
DS7	Text	98.38	–	97.43	86.76	**96.07**
	Equation	98.74	–	–	–	
	Graphical	97.73	97.38	–	–	

and *module* be the performance measurement, and a set of modules presented in DCS, then its performance against dataset D can be measured as

$$\Theta(DCS_D) = \frac{1}{M}\sum_{i=1}^{M}\Theta(module_i) \tag{A.1}$$

where M is the number of modules present in DCS which are active for the evaluation with respect to the dataset D.

Table A.6 presents the performance of the DCS (in accuracy) with all the performance provided by each module depending on the type of dataset. From the table, the following observations are made:

1. With respect to the module **Document segmentation and classification**, it receives its highest accuracy of 99.02% over dataset DS1, which consists of only textual components, and accrued the lowest accuracy of only 90.93% over dataset DS2, which consists of document images with only graphical regions. However, over the dataset DS7, which is made up of document images with all three components, it accrued an average accuracy of 98.23%. Its lowest segmentation and

identification are reported over dataset DS4, which consists of only textual and mathematical equations. It is because of the presence of an enormous amount of text-line-alike equation components.

2. With respect to the module **Chart type classification**, it receives an average accuracy of 98% over four datasets that are applicable for this module, namely DS3, DS5, DS6, and DS7. Therefore, this module provides comparatively better performance with no respect to the nature of the dataset. It is because its input is not the entire document image but a segment which are identified as graphical by the previous module.

3. The performance of the module **Manipuri OCR** is reported for four datasets that have textual components (not the embedded textual but regular textual). It is observed that its average performance over four datasets, namely DS1, DS4, DS5, and DS6, is 97.03%. Its lowest accuracy of 96.32% is observed over the dataset DS4 which consists of only textual and mathematical components. It's dropping in performance over this dataset is because of the inaccurate segmentation and identification of mathematical equations and textual components.

4. The performance of the module **Bengali-Meetei script transliteration** is the lowest among all modules. On average, it provides an accuracy of 86.02%.

5. Finally, the performance of **DCS** is calculated for the respective dataset with the expression in equation A.1. The poor performance of the module, Machine transliteration, affects the overall achievement of DCS. On average, considering all seven dataset types, DCS accrued an accuracy of 95.78%.

Figure A.4 shows the structurally transliterated output samples of DCS over the document image samples in dataset DS5. Fitting of graphical or equation regions (which are in image format) in their respective positions sometimes compresses the regions because

of the extra pixels segmented with the regions during the segmentation process. Figure A.5 presents the generated meta-file by DCS. For a given input document image x, the generated meta-file, x' generated by DCS could be demonstrated as $x' = \{s_1, s_2 ... s_k, \delta, \mu, \lambda\}$, where k is the number of segmented regions or bounding boxes obtained for the input image, x. The s_i can be defined as $\{\rho_i, \pi_i, \gamma_i, \tau_i\}$, where

- ρ_i denotes the coordinates of the bounding box s_i
- π_i denotes the class of the bounding box s_i: Text, Equation, Graphical
- γ_i denotes the category of the graphical region. So, it is NULL for the s_i if its class is either Equation or Textual.
- τ_i is a text file that contains textual components if s_i is textual region.. So, it contains NULL for the s_i if its class is either Equation or Graphical.

The remaining parameters in x', namely δ_i, ζ_i, and λ_i denote the number of words, number of text lines, and number of graphical objects, respectively, in the input document x . Therefore, with DCS, for any given Manipuri Bengali document image, it provides a PDF document transliterated to Meetei script and a JSON file that contains the meta-information for the given input image.

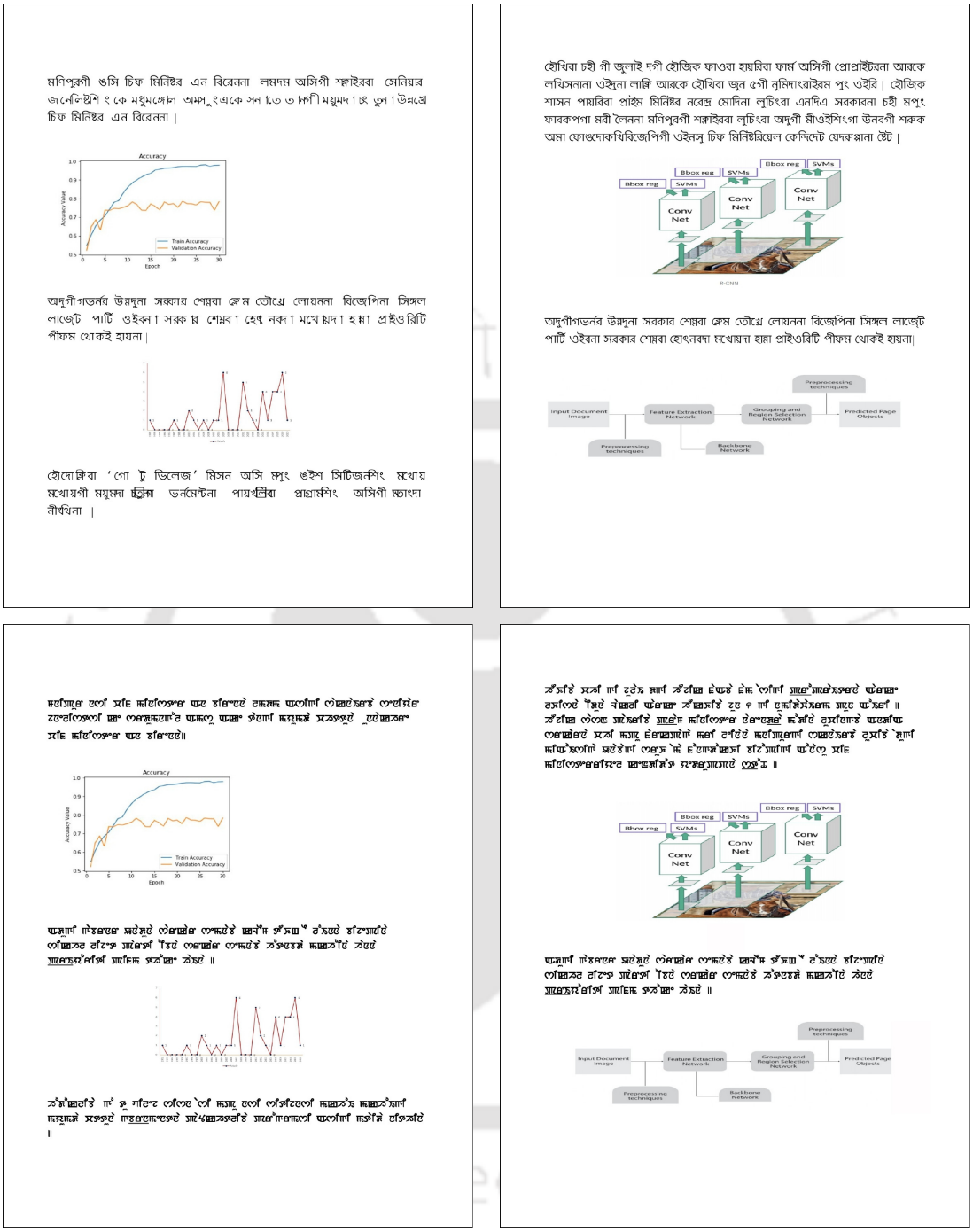


Figure A.4: Transliterated output samples of DCS : Output samples in Meetei script (in the second row) for the corresponding input document samples in Bengali script (in the first row).

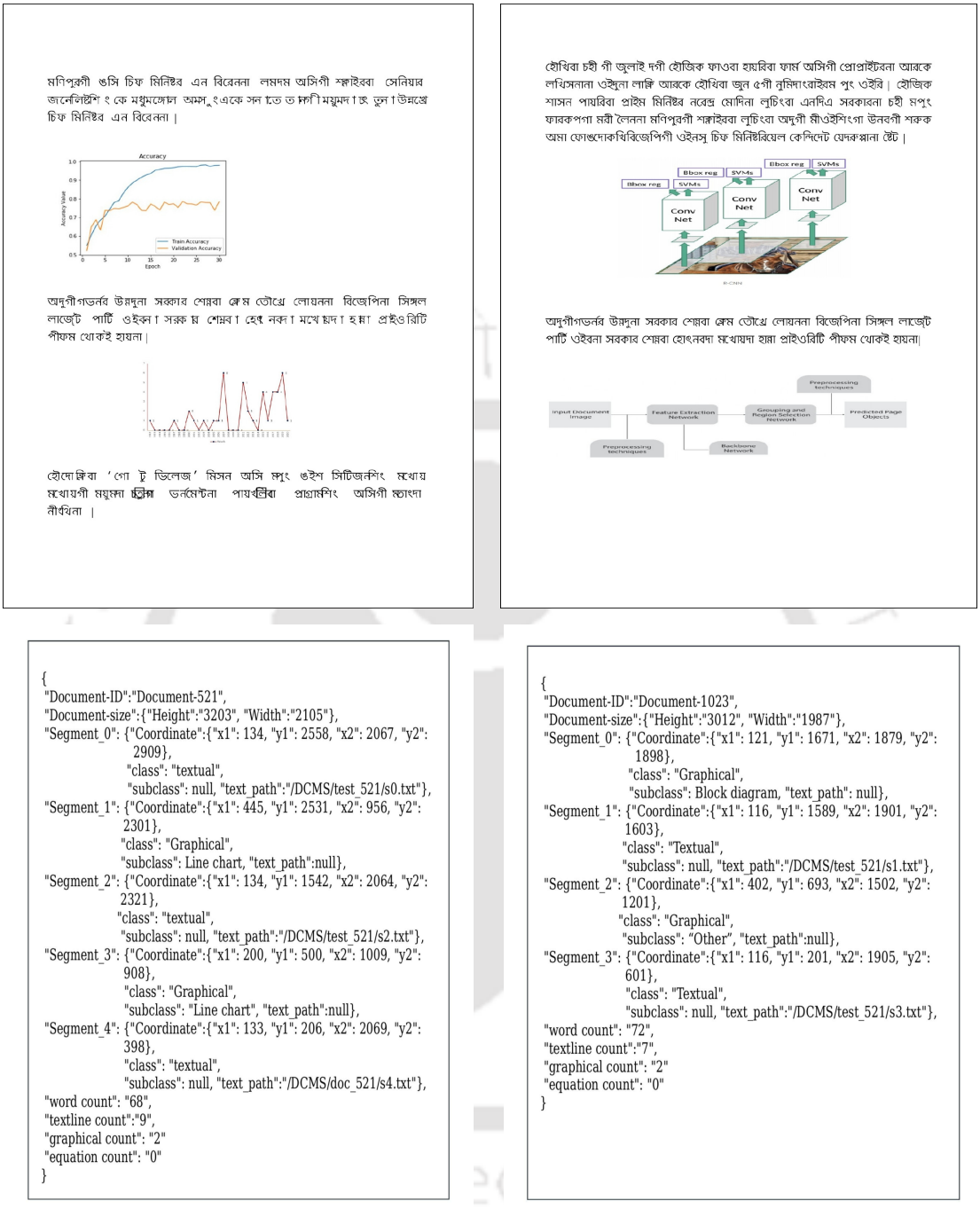


Figure A.5: Output samples of meta-file generated by DCS : Meta-file samples (in the second row) produced by DCS for the corresponding input samples (in the first row).

B

Appendix

B.1 Some well known CNN models

Convolutional Neural Networks (CNNs or ConvNets) are specialized deep learning architectures that excel in analyzing image content. They have garnered significant attention from major tech companies like Google, Microsoft, and Facebook, which have actively invested in research groups dedicated to exploring new CNN architectures. These companies have showcased the effectiveness of CNNs in various computer vision tasks, including image segmentation, classification, detection, and retrieval. Over time, nu-

merous variants of CNN architectures have been developed to address real-world challenges. These architectures are designed to enhance the performance and capabilities of CNNs in tackling complex visual understanding problems. This thesis provides a concise overview of four well-known CNN models, highlighting their distinct approaches to feature extraction, hierarchical representation learning, and information processing.

B.1.1 LeNet

LeNet-5 is widely recognized as one of the most prominent convolutional neural networks (CNNs) ever developed. It consists of five layers, excluding pooling, making it a compact architecture for its time. The network architecture comprises two convolutional layers, each utilizing a kernel size of 5×5, followed by three fully connected layers. In LeNet-5, each convolutional layer is succeeded by a 2×2 average pooling operation, which helps reduce spatial dimensions while retaining important features. The activation function used in every layer, except the last one, is the hyperbolic tangent (tanh). The final layer of the network employs the softmax activation function, enabling the classification of input images into one of ten digit classes (0 to 9).

With a total of 60,000 parameters, LeNet-5 is relatively lightweight compared to modern CNN architectures. It was originally trained on grayscale images of size 32×32, with the objective of recognizing and classifying handwritten digits.

B.1.2 AlexNet

AlexNet introduced significant enhancements to CNN architecture, including the adoption of the Rectified Linear Unit (ReLU) activation function and Local Response Normalization (LRN). ReLU gained widespread popularity and became a standard choice for subsequent CNN designs, replacing the use of the tanh activation function in LeNet-5. It consists of eight layers, including three fully connected layers and five convolutional

layers with decreasing kernel sizes. ReLU is employed in all layers except the final layer, which utilizes the softmax activation function for classification purposes. Additionally, LRN is applied to the first and second convolutional layers. Max pooling with a 3×3 window is performed after the first, second, and fifth convolutional layers.

With advancements in hardware capabilities, the modern implementation of AlexNet can be trained with a staggering 60 million parameters. The network achieved notable success by winning the ImageNet competition in 2012. ImageNet, a widely-used benchmark dataset, comprises a diverse collection of images classified into 1,000 different classes. It is typically trained on colored images with dimensions of 224×224, reflecting the default input size for this architecture.

B.1.3 VGG-16

In the pursuit of understanding the impact of CNN depth on large-scale image recognition, researchers explored the effect of increasing the number of layers within the architecture. This investigation led to the creation of four VGG model variations: VGG-11, VGG-13, VGG-16, and VGG-19, with depths ranging from 11 to 19 layers. Initially, a version of VGG-11 incorporating Local Response Normalization (LRN) was examined, but it was found that LRN did not enhance performance. As a result, all subsequent VGG models were implemented without LRN.

VGG-16, one of the VGG families, is composed of 16 layers, specifically 13 convolutional layers with a kernel size of 3×3, followed by three fully connected layers. This network stands out as one of the larger architectures, featuring a staggering 138 million parameters. Similar to AlexNet, the final layer of VGG-16 employs the softmax activation function for classification, while all other layers utilize the Rectified Linear Unit (ReLU) activation function. Max-pooling with a 2×2 window is performed after the second, fourth, seventh, tenth, and thirteenth convolutional layers. By default, VGG-16 is

designed to process colored images with dimensions of 224×224 and output predictions among 1,000 distinct classes. Its depth and parameter count make it a substantial and effective network for image recognition tasks, contributing to advancements in the field of computer vision.

B.1.4 ResNet-50

When deeper networks are utilized, a phenomenon known as degradation occurs, where the accuracy of the network initially saturates and then rapidly deteriorates. Interestingly, this degradation is not due to overfitting, as increasing the depth of a suitably deep network leads to higher training error rather than lower training error and higher testing error.

To address this degradation issue, the concept of bottleneck residual blocks was introduced. These blocks aim to overcome the deterioration problem and improve the performance of deep networks.

- Identity block: This block consists of three convolutional layers with kernel sizes of 1×1, 3×3, and 1×1, respectively. Batch Normalization (BN) is applied to all three layers. The ReLU activation function is employed for the first two layers, while the input of the identity block is added to the output of the last layer before applying ReLU.

- Convolution block: Similar to the identity block, the convolution block also comprises three convolutional layers with the same kernel sizes. However, before being added to the last convolutional layer of the main series, the input of the convolution block first passes through a convolution layer with a 1×1 kernel size and BN.

Each residual block, whether an identity block or a convolution block, consists of

three layers in total. ResNet-50, a popular variant, is a deep network architecture that includes 50 layers and a total of 26 million parameters. By default, ResNet-50 is designed to output predictions among 1,000 classes and accepts colored images with dimensions of 224×224. The introduction of residual blocks, such as those found in ResNet-50, has proven effective in combating the degradation issue and has significantly contributed to the success of deep network architectures in various computer vision tasks.

B.1.5 Inception-v1

Exploding/vanishing gradients are one result of deeperening a network. When significant error gradients build up and cause erratic weight updates during training, the gradient explodes. The vanishing gradient, on the other hand, becomes an issue when the partial derivative of the loss function gets near to zero and the network is unable to train. Inception-v1 attempts to address this problem by incorporating two auxiliary classifiers linked to intermediate layers, with the goal of boosting the gradient signal that is returned. Their loss is combined with the network's overall loss during training with a discount weight of 0.3. These auxiliary networks are discarded at the moment of inference. Inception-v1 adds the inception module, which consists of four simultaneous stacks of one or two convolution layers and one or more max-pool layers. The inception module, which permits the use of various kernel sizes rather than being limited to a single kernel size, seeks to simulate an ideal local sparse structure in a CNN. A mere 7 million parameters separate Inception-v1 from AlexNet and VGG-16, despite the fact that it has 22 layers total: three convolution layers with 77, 11, and 33 kernel sizes, followed by 18 layers made up of 9 Inception modules, each of which has two layers of convolution/max-pooling, and one fully connected layer. All additional layers are equipped with ReLU, with the exception of the final layer of the main classifier and the two auxiliary classifiers. A 33 max-pooling is applied after the first and third convolu-

tion layers, as well as the second and seventh inception modules. A 77 average-pooling is followed by the final inception module. After the first max-pooling and the third convolution layer, LRN is applied. After the third and sixth inception modules, auxiliary classifiers are branched out. Each begins with a 55 average-pooling and is followed by three layers: one convolution layer with an 11 kernel size, and two fully linked layers. By default, Inception-v1 outputs one of the 1000 classes and accepts coloured images with a 224x224 size.

B.1.6 Xception

Xception, short for "extreme inception," takes the concepts introduced by the Inception architecture to the next level. In Inception, the input was compressed using 1x1 convolutions, and then different filters were applied to each compressed input space. However, Xception reverses this process. It applies filters to each depth map individually before utilizing 1x1 convolutions to reduce the input space across the depth maps. This approach can be seen as a form of depthwise separable convolution, which was first introduced in neural network architectures in 2014.

The key distinction between Inception and Xception lies in the presence or absence of a non-linearity after the initial operation. In the Inception model, a Rectified Linear Unit (ReLU) non-linearity follows both processes, ensuring non-linear transformations. On the other hand, Xception does not include any additional non-linearity after the initial operations. By leveraging this unique approach, Xception achieves a more efficient and expressive representation of the input data. Standard Xception models are designed to output predictions among 1,000 classes and accept colored images with dimensions of 299x299 pixels. It showcases advancements in convolutional neural networks and highlights the importance of depthwise separable convolutions in reducing computational complexity while maintaining strong representational power. Its utilization of depth-

wise separable convolutions and absence of intermediate non-linearities contribute to its distinctive characteristics and success in various computer vision tasks. 4

B.2 Tesseract OCR

This section develops the developing procedure of OCR from existing OCR using Tesseract. As this thesis is focused on developing Manipuri Bengali OCR, the explanation of Tesseract is done from the perspective of developing Tesseract-based Manipuri OCR from the existing tesseract based Bengali OCR.

During the period from 1985 to 1994, OCRTesseract was developed collaboratively by Hewlett-Packard Laboratories in Bristol and Hewlett-Packard Company in Greeley, Colorado. In 1996, it was ported to Windows, and in 1998, it underwent some civilization. Tesseract was made publicly available in 2005 by HP and the University of Nevada, Las Vegas (UNLV). It has been created by Google since 2006. The most recent stable version (LSTM-based) is 4.0.0, which was released on October 29, 2018.3.

B.2.1 Why Tesseract – OCR?

Tesseract is a popular optical character recognition (OCR) engine known for its extensive language support, including robust handling of Unicode (UTF-8) encoding. With Tesseract, users can recognize text in over 100 languages without the need for additional configuration. Moreover, Tesseract provides the flexibility to train the engine to recognize additional languages as needed. It offers various output formats such as plain text, hOCR (HTML), PDF, invisible-text-only PDF, and TSV. Additionally, the experimental support for ALTO (XML) output is available in the master branch of Tesseract. Notably, Tesseract is highly regarded for its remarkable accuracy, setting it apart from other OCR engines.

B.2.2 Installing Tesseract

To install Tesseract 4.x on Ubuntu 18.xx, you can easily execute the following command:

- sudo apt install tesseract-ocr

To install the Developer Tools required for training purposes, you can use the following command:

- sudo apt install libtesseract-dev

B.2.3 Fine Tune

Fine-tuning is a feature offered by the Tesseract engine, allowing users to start with a pre-trained language model and further train it using their own specific additional data. This approach is particularly effective when dealing with problems that are similar to the existing training data but have some subtle differences, such as unique fonts or specific variations. Fine-tuning can yield good results even with a small amount of training data. In Tesseract 4.xx, there are two main types of fine-tuning available.

- Fine tuning for specific fonts.
- Fine tuning for additional symbols.

B.2.4 Requirement in Linux system

- Tools:
  - Tesseract 4.1.0-rc
  - GPL Ghostscript 9.26 (2018-11-20)
  - OCR-evaluation-Tools
- Dependencies:

- A compiler for C and C++: GCC or Clan
- GNU Autotools: autoconf, automake, libtool
- pkg-config
- Leptonica
- libpng, libjpeg, libtif

B.2.5 Data

- *ben.traineddata:* This is Bengali OCR model provided by Tesseract.
- *langdata_lstm_master:* This contains the training text for all the languages/scripts used by the tesseract to developed their respective OCR models (ben.training_txt for Bengali OCR, ben.traineddata).

B.2.6 Modification of *ben.training_txt*

The actual training text, ben.training_txt is modified to include the characters or words or textline we want to include. So the Manipuri Bengali characters are included and fine tune the model ben.traineddat to recognized Manipuri bengali script which it has never seen before. So, the ben.training_txt is updated by appending the Manipuri Bengali words or texlines.

B.2.7 Running tesstrain.sh

Training data is created using tesstrain.sh as follows:

src/training/tesstrain.sh –fonts_dir /usr/share/fonts –lang ben
–linedata_only
–noextract_font_properties –langdata_dir ../langdata

–fontlist Bangla Medium Lohit Bengali

–tessdata_dir ./tessdata –output_dir /tesstutorial/ben_man

where “Bangla Medium” “Lohit Bengali” are the font style in which the fine-tunbing is hoing to be done. The next step is to run tesstrain.sh on the modified ben.training_txt using the font we want to train. It produces an output with five files as follows

- ben.font.exp0.lstmf
- ben.training_files.txt
- ben.charset_size=xyz.txt
- ben.traineddata
- ben.unicharset

Running tesstrain.sh also generate tiff and box files from the training_txt.

B.2.8 Fine Tuning on the new training data

In tesseract-ocr 4.xx we can fine tune a existing model using:

- **training/combine_tessdata -e tessdata/lang.traineddata/lang.lstm**

- **training/lstmtraining –model_output /path/to/output**
  –continue_from /path/to/existing/model
  –traineddata /path/to/traineddata/with/new/unicharset
  –old_traineddata /path/to/existing/traineddata
  –perfect_sample_delay 0 –debug_interval 0

–max_iterations 0 –target_error_rate 0.01

–train_listfile /path/to/list/of/filenames.txt

The above two steps of commands will provide the output as model_name.checkpoint.The flag with its default values:

- –debug_interval 0 , dring the training process, the trainer will provide a progress report every 100 iterations.
- –perfect_sample_delay 0, allows for the discarding of perfect samples if there haven't been a sufficient number of imperfect samples seen since the last perfect sample. The default value of zero includes all samples, and in practice, this value doesn't have a significant impact. If the training is allowed to run for an extended period, using zero as the value tends to yield the best results.
- –target_error_rate 0.01, training will be stopped if the mean percent error rate falls below a certain threshold value.
- –max_iterations 0, Stop training after this many iterations.

The final step of fine tuning is to combine data files. The lstmtraining program outputs two kinds of checkpoint files:

- <model_base>_checkpoint is the latest model file.
- <model_base><char_error>_<iteration>.checkpoint is periodically written as the model with the best training error. It is a training dump just like the checkpoint, but is smaller because it doesn't have a backup model to be used if the training runs into divergence.

Either of these files can be converted to a standard traineddata file as follows:

training/lstmtraining –stop_training

–continue_from /tesstutorial/ben_from_chi/base_checkpoint

–traineddata /tesstutorial/bentrain/ben/ben.traineddata

–model_output /tesstutorial/ben_from_chi/ben.traineddata

This process involves extracting the recognition model from the training dump and incorporating it into the –traineddata argument, along with the unicharset, recoder, and any dawgs that were used during the training process.

References

[1] Akram, Q. U. A. & Hussain, S. (2017). Ligature-based font size independent OCR for Noori Nastalique writing style. In 1st International Workshop on Arabic Script Analysis and Recognition (ASAR) (pp. 129–133). Nancy, France: IEEE.

[2] Amara, J., Kaur, P., Owonibi, M., & Bouaziz, B. (2017). Convolutional Neural Network based chart image classification. In Proceedings of 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (pp. 111–119). Plzen, Czech Republic: Vaclav Skala Union Agency(VSUA).

[3] Antonacopoulos, A. & Ritchings, R. (1995). Representation and classification of complex-shaped printed regions using white tiles. In Proceedings of 3rd international conference on document analysis and recognition, volume 2 (pp. 1132–1135). Montreal, Canada: IEEE.

[4] Anwar, N., Khan, T., & Mollah, A. F. (2022). Text detection from scene and born images: How good is Tesseract? In Recent Trends in Communication and Intelligent Systems (pp. 115–122). Singapore: Springer.

[5] Ashwin, T. & Sastry, P. (2002). A font and size-independent OCR system for printed Kannada documents using support vector machines. Sadhana, 27, 35–58.

[6] Baird, H. S., Jones, S. E., & Fortune, S. J. (1990). Image segmentation by shape-directed covers. In Proceedings of [1990] Proceedings. 10th International Conference on Pattern Recognition, volume 1 (pp. 820–825). Atlantic City, NJ, US: IEEE.

[7] Bajić, F., Job, J., & Nenadić, K. (2019). Chart classification using simplified VGG model. In Proceedings of 2019 International Conference on Systems, Signals and Image Processing (IWSSIP) (pp. 229–233). Osijek, Croatia: IEEE.

[8] Balaji, A., Ramanathan, T., & Sonathi, V. (2018). Chart-text: A fully automated chart image descriptor. arXiv preprint arXiv:1812.10636.

[9] Bhattacharyya, K. & Sarma, K. (2009). Innovative segmentation of handwritten text in Assamese using Neural Network. In Proceedings of the International Conference on Image Processing, Computer Vision, Pattern Recognition, IPCV (pp. 585–590). Las Vegas, Nevada, USA.

[10] Björkman, J. (2019). Evaluation of the effects of different preprocessing methods on OCR results from images with varying quality. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS).

[11] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5–32.

[12] Breuel, T. M. (2017). High performance text recognition using a hybrid Convolutional-LSTM implementation. In Proceedings of 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 01 (pp. 11–16). Kyoto, Japan: IEEE.

[13] Brisinello, M., Grbić, R., Pul, M., & Anđelić, T. (2017). Improving optical character recognition performance for low quality images. In 2017 International Symposium ELMAR (pp. 167–171). Zadar, Croatia: IEEE.

[14] Bukhari, S., Azawi, M., Shafait, F., & Breuel, T. (2010). Document image segmentation using discriminative learning over connected components. In Proceedings of ACM International Conference Proceeding Series (pp. 183–190). Boston Massachusetts USA: ACM.

[15] Chagas, P., Akiyama, R., Meiguins, A., Santos, C., Saraiva, F., Meiguins, B., & Morais, J. (2018). Evaluation of Convolutional Neural Network architectures for chart image classification. In International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). Rio, Brazil: IEEE.

[16] Chagas, P., Freitas, A., Daisuke, R., Miranda, B., Araújo, T. D. O. D., Santos, C., Meiguins, B., & Morais, J. M. D. (2017). Architecture proposal for data extraction of chart images using Convolutional Neural Network. In Proceedings of 21st International Conference Information Visualisation (IV) (pp. 318–323). London, United Kingdom: IEEE.

[17] Chaudhuri, B. B. & Pal, U. (1997). An OCR system to read two indian language scripts: Bangla and Devnagari (Hindi). volume 2 (pp. 1011–1015). Ulm, GERMANY: IEEE.

[18] Chaudhuri, B. B. & Pal, U. (1998). A complete printed Bangla OCR system. Pattern Recognition, 31, 531–549.

[19] Che, E., Jung, J., & Olsen, M. J. (2019). Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. Sensors, 19(4).

[20] Chen, B.-C., Chen, C.-S., & Hsu, W. H. (2015). Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset. IEEE Transactions on Multimedia, 17(6), 804–815.

[21] Chen, K., Yin, F., & Liu, C.-L. (2013). Hybrid page segmentation with efficient whitespace rectangles extraction and grouping. In Proceedings of 12th International Conference on Document Analysis and Recognition (pp. 958–962). Washington, DC, USA: Springer.

[22] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1800–1807). Honolulu, HI, USA: IEEE.

[23] Clark, C. & Divvala, S. (2016). PDFFigures 2.0: Mining figures from research papers. In IEEE/ACM Joint Conference on Digital Libraries (JCDL) (pp. 143–152). Newark, NJ, USA: IEEE.

[24] Clark, C. A. & Divvala, S. (2015). Looking beyond text: Extracting figures, tables and captions from computer science papers. In Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence Austin Texas, USA: AAAI.

[25] Cover, T. & Hart, P. (1981). A lazy learning approach to classification: K-nearest neighbor. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 13(4), 328–335.

[26] Cui, Y., Zhou, F., Lin, Y., & Belongie, S. J. (2016). Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (pp. 1153–1162).

[27] Dai, W., Wang, M., Niu, Z., & Zhang, J. (2018). Chart Decoder generating textual and numeric information from chart images automatically. Journal of Visual Languages Computing, 48, 101–109.

[28] Dalal, N. & Triggs, B. (2005a). Histograms of oriented gradients for human detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 886–893). San Diego, CA, USA: IEEE.

[29] Dalal, N. & Triggs, B. (2005b). Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1 (pp. 886–893). San Diego, California: IEEE.

[30] Das, A., Ekbal, A., Mondal, T., & Bandyopadhyay, S. (2009). English to Hindi machine transliteration system at NEWS 2009. In Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009) (pp. 80–83). Suntec, Singapore: ACL.

[31] Davila, K., Kota, B. U., Setlur, S., Govindaraju, V., Tensmeyer, C., Shekhar, S., & Chaudhry, R. (2019). ICDAR 2019 competition on harvesting raw tables from infographics (chart-infographics). In Proceedings of International Conference on Document Analysis and Recognition (ICDAR) (pp. 1594–1599). Sydney: IEEE.

[32] Davila, K., Tensmeyer, C., Shekhar, S., Singh, H., Setlur, S., & Govindaraju, V. (2021). ICPR 2020-competition on harvesting raw tables from infographics. In Proceedings of International Conference on Pattern Recognition (pp. 361–380). Milano, Italy: Springer.

[33] Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA transactions on Signal and Information Processing, 3, 1–29.

[34] Domingos, P. & Pazzani, M. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Artificial Intelligence Research, 2, 337–369.

[35] Ekbal, A., Naskar, S., & Bandyopadhyay, S. (2007). Named entity transliteration. International Journal of Computer Processing of Languages, 20, 289–310.

[36] Ekbal, A., Naskar, S. K., & Bandyopadhyay, S. (2006). A modified joint source-channel model for transliteration. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions (pp. 191–198). Sydney, Australia: ACL.

[37] Eskenazi, S., Gomez-Krämer, P., & Ogier, J. M. (2017). A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognition, 64(June 2020), 1–14.

[38] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. International Journal of Computer Vision, 88(2), 303–338.

[39] Filip Bajić, Josip Job, K. N. (2020). Data visualization classification using simple Convolutional Neural Network model. International Journal of Electrical and Computer Engineering Systems, 11, 43–51.

[40] Fletcher, L. & Kasturi, R. (1988). A robust algorithm for text string separation from mixed text/graphics images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(6), 910–918.

[41] Futrelle, R., Shao, M., Cieslik, C., & Grimes, A. (2003). Extraction,layout analysis and classification of diagrams in PDF documents. In Proceedings of Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. (pp. 1007–1013).

[42] Futrelle, R. P., Kakadiaris, I. A., Alexander, J., Carriero, C. M., Nikolakis, N., & Futrelle, J. M. (1992). Understanding diagrams in technical documents. Computer, 25(7), 75–78.

[43] Ganapathiraju, M., Mini, B., Balakrishnan, N., & Raj, R. (2005). Om: One tool for many (indian) languages. Journal of Zhejiang University - Science A: Applied Physics Engineering, 6, 1348–1353.

[44] Gao, J., Zhou, Y., & Barner, K. E. (2012). View: Visual information extraction widget for improving chart images accessibility. In Proceedings of 19th IEEE International Conference on Image Processing (pp. 2865–2868). Florida, U.S.A.: IEEE.

[45] Gao, L., Yi, X., Liao, Y., Jiang, Z., Yan, Z., & Tang, Z. (2017). A deep learning-based formula detection method for PDF documents. In Proceedings of 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 01 (pp. 553–558). Kyoto, Japan: Springer.

[46] Garg, A. & Mago, V. (2021). Role of machine learning in medical research: A survey. Computer Science Review, 40, 100370.

[47] Ghosh, S., Bora, P. K., Das, S., & Chaudhuri, B. B. (2012). Development of an assamese OCR using Bangla OCR. In Workshop on Document Analysis and Recognition (pp. 68–73). New York, NY, USA: ACM.

[48] Giannakopoulos, T., Foufoulas, I., Stamatogiannakis, E., Dimitropoulos, H., Manola, N., & Ioannidis, Y. (2015). Visual-based classification of figures from scientific literature. In Proceedings of the 24th International Conference on World Wide Web (pp. 1059–1060). New York, NY, USA: ACM.

[49] Gilani, A., Qasim, S. R., Malik, I., & Shafait, F. (2017). Table detection using deep learning. In Proceedings of 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 01 (pp. 771–776). Kyoto, Japan: Springer.

[50] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. Journal of Machine Learning Research, 15, 315–323.

[51] Guo, S., Wang, S., Guo, J., & Xu, J. (2021). Classification of aquatic animals by the spherical amphibian robot based on transfer learning. In Proceedings of IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 1213–1218). Takamatsu, Kagawa, Japan: IEEE.

[52] Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 2 (pp. 1735–1742).

[53] Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610–621.

[54] He, G., Li, F., Wang, Q., Bai, Z., & Xu, Y. (2021). A hierarchical sampling based triplet Network for fine-grained image classification. Pattern Recognition, 115, 107889.

[55] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770–778). Salt Lake City, UT, USA: IEEE.

[56] Hearst, M., Dumais, S., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their Applications, 13(4), 18–28.

[57] Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527–1554.

[58] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient Convolutional Neural Networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[59] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation Networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132–7141). Salt Lake City, UT, USA: IEEE.

[60] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected Convolutional Networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700–4708). Honolulu, HI, USA: IEEE.

[61] Huang, W. & Tan, C. L. (2007). A system for understanding imaged infographics and its applications. In Proceedings of the 2007 ACM Symposium on Document Engineering, DocEng '07 (pp. 9–18). New York, NY, USA: Association for Computing Machinery.

[62] Huang, W., Zong, S., & Tan, C. L. (2007). Chart image classification using multiple-instance learning. In IEEE Workshop on Applications of Computer Vision (WACV'07) (pp. 27–27). Waikoloa, Hawaii: IEEE.

[63] Ibrahim, A. (2014). Dhivehi OCR: Character recognition of Thaana script using machine generated text and Tesseract OCR engine. Maldives Journal of Research, 1.

[64] Jain, M., Mathew, M., & Jawahar, C. V. (2017). Unconstrained OCR for Urdu using deep CNN-RNN hybrid Networks. In 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR) (pp. 747–752). Nanjing, China.

[65] Javed, S. T., Hussain, S., Maqbool, A., Asloob, S., Jamil, S., & Moin, H. (2010). Segmentation free Nastalique Urdu OCR. International Journal of Computer and Information Engineering, 4(10), 1514 – 1519.

[66] Jobin, K. V., Mondal, A., & Jawahar, C. V. (2019). DocFigure: A dataset for scientific document figure classification. In ICDARW, volume 1 (pp. 74–79).

[67] Jung, D., Kim, W., Song, H., Hwang, J.-i., Lee, B., Kim, B., & Seo, J. (2017). ChartSense: Interactive Data Extraction from Chart Images, (pp. 6706–6717). ACM: New York, NY, USA.

[68] Kae, A. & Learned-Miller, E. (2009). Learning on the Fly: Font-free approaches to difficult OCR problems. In Proceedings of 10th International Conference on Document Analysis and Recognition (pp. 571–575). Barcelona, Spain: IEEE.

[69] Kang, K., Pang, G., Zhao, X., Wang, J., & Li, Y. (2020). A new benchmark for instance-level image classification. IEEE Access, 8, 70306–70315.

[70] Kang, L., Kumar, J., Ye, P., Li, Y., & Doermann, D. (2014). Convolutional Neural Networks for document image classification. In Proceedings of 22nd International Conference on Pattern Recognition (pp. 3168–3172). Stockholm, Sweden: IEEE.

[71] Karis, M. S., Razif, N. R. A., Ali, N. M., Rosli, M. A., Aras, M. S. M., & Ghazaly, M. M. (2016). Local binary pattern (LBP) with application to variant object detection: A survey and method. In IEEE 12th International Colloquium on Signal Processing & Its Applications (CSPA) (pp. 221–226). Melaka, Malaysia: IEEE.

[72] Karthikeyani, V. & Nagarajan, S. (2011). Scientific chart image property identification by connected component labeling in PDF files. In Proceedings of 3rd International Conference on Electronics Computer Technology, volume 4 (pp. 209–212). Kanyakumari, India: IEEE.

[73] Karthikeyani, V. & Nagarajan, S. (2012). Machine learning classification algorithms to recognize chart types in portable document format (PDF) files. International Journal of Computer Applications, 39, 1–5.

[74] Kavasidis, I., Pino, C., Palazzo, S., Rundo, F., Giordano, D., Messina, P., & Spampinato, C. (2019). A saliency-based Convolutional Neural Network for table and chart detection in digitized documents. In Proceedings of International conference on image analysis and processing (pp. 292–302). Italy: Springer.

[75] Khanuja, S., Bansal, D., Mehtani, S., Khosla, S., Dey, A., Gopalan, B., Margam, D. K., Aggarwal, P., Nagipogu, R. T., Dave, S., et al. (2021). Muril: Multilingual representations for indian languages. arXiv preprint arXiv:2103.10730.

[76] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90.

[77] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep Convolutional Neural Networks. Communicarion of ACM, 60(6), 84–90.

[78] Kulkarni, M., Karande, S. S., & Lodha, S. (2016). Unsupervised word clustering using deep features. In 12th IAPR Workshop on Document Analysis Systems (DAS) (pp. 263–268). Santorini, Greece: IEEE.

[79] Kumar, R., Weill, E., Aghdasi, F., & Sriram, P. (2019). Vehicle re-identification: an efficient baseline using triplet embedding. International Joint Conference on Neural Networks (IJCNN), (pp. 1–9).

[80] Kumar, S. S., Rajendran, P., Prabaharan, P., & Soman, K. (2016). Text/image region separation for document layout detection of old document images using non-linear diffusion and level set. Procedia Computer Science, 93, 469–477.

[81] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

[82] Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z., & Zhou, M. (2020). Docbank: A benchmark dataset for document layout analysis. arXiv preprint arXiv:2006.01038.

[83] Lin, M.-W., Tapamo, J.-R., & Ndovie, B. (2006). A texture-based method for document segmentation and classification. South African Computer Journal, 2006(36), 49–56.

[84] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110.

[85] Malik, M. G. A. (2006). Punjabi machine transliteration. In Proceedings of 21st international Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the ACL (pp. 1137–1144). USA: ACL.

[86] Manisha, U. & Liyanage, S. (2018). Sinhala character recognition using Tesseract OCR. In Proceedings of 3rd International Conference on Advances in Computing and Technology (ICACT) Sri Lanka: Faculty of Computing and Technology, University of Kelaniya.

[87] Mathew, M., Singh, A. K., & Jawahar, C. (2016). Multilingual OCR for Indic scripts. In 12th IAPR Workshop on Document Analysis Systems (DAS) (pp. 186–191). Santorini, Greece: IEEE.

[88] Medsker, L. R. & Jain, L. (2001). Recurrent Neural Networks. Design and Applications, 5, 64–67.

[89] Mishchenko, A. & Vassilieva, N. (2011). Model-based recognition and extraction of information from chart images. In Journal of Multimedia Processing and Technologies, volume 2 (pp. 76–89).

[90] Mori, S., Suen, C., & Yamamoto, K. (1992). Historical review of OCR research and development. Proceedings of the IEEE, 80(7), 1029–1058.

[91] Mujumdar, S., Gupta, N., Jain, A., & Burdick, D. (2019). Simultaneous optimisation of image quality improvement and text content extraction from scanned documents. In Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR) (pp. 1169–1174). Sydney, Australia, Australia: IEEE.

[92] Mulgaonkar, P. G., Chen, C.-H., & DeCurtins, J. L. (1994). Word recognition in a segmentation-free approach to OCR. In J. M. Selander (Ed.), 22nd AIPR Workshop: Interdisciplinary Computer Vision: Applications and Changing Needs, volume 2103 (pp. 135 – 141). Tsukuba Science City, Japan: International Society for Optics and Photonics SPIE.

[93] N, K. (2010). A comparative study of Meetei Mayek: from the inscribed letterform to the digital typeface. Unpublished Masters Dissertation. University of Reading. Reading, UK.

[94] Naseer, A. & Zafar, K. (2018). Comparative analysis of raw images and meta feature based Urdu OCR using CNN and LSTM. volume 9 (pp. 419–424).

[95] Nayak, M. & Kumar, A. (2013). Odia characters recognition by training Tesseract OCR engine. International Journal of Computer Applications, (pp. 975–8887).

[96] Nongmeikapam, K., Singh, N. H., Thoudam, S., & Bandyopadhyay, S. (2011). Manipuri transliteration from bengali script to meitei Mayek: A rule based approach. In Proceedings of International Conference on Information Systems for Indian Languages (pp. 195–198). Berlin, Heildelberg, Germany: Springer.

[97] Ojala, T., Pietikäinen, M., & Harwood, D. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In Proceedings of the 16th International Conference on Pattern Recognition (ICPR) (pp. II–II). Quebec City, Canada: IEEE.

[98] Olga Russakovsky, Jia Deng, H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. h.-O. L. F.-F. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115, 211–252.

[99] Oyedotun, O. K. & Khashman, A. (2016). Document segmentation using textural features summarization and feedforward Neural Network. Applied Intelligence, 45(1), 198–212.

[100] Paliwal, S. S., D, V., Rahul, R., Sharma, M., & Vig, L. (2019). TableNet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In Proceedings of International Conference on Document Analysis and Recognition (ICDAR) (pp. 128–133). International Conference on Document Analysis and Recognition (ICDAR): IEEE.

[101] Paul, D. & Chaudhuri, B. B. (2019). A BLSTM Network for printed bengali OCR system with high accuracy. arXiv preprint arXiv:1908.08674.

[102] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[103] Poco, J. & Heer, J. (2017). Reverse-Engineering Visualizations: Recovering visual encodings from chart images. Computer Graphics Forum, 36, 353–363.

[104] Pondenkandath, V., Seuret, M., Ingold, R., Afzal, M. Z., & Liwicki, M. (2017). Exploiting state-of-the-art deep learning methods for document image analysis. In Proceedings of 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 05 (pp. 30–35). Kyoto, Japa: Springer.

[105] Prasad, V. S. N., Siddiquie, B., Golbeck, J., & Davis, L. S. (2007). Classifying computer generated charts. In International workshop on content-based multimedia indexing (pp. 85–92). Bordeaux, France: IEEE.

[106] Qaroush, A., Jaber, B., Mohammad, K., Washaha, M., Maali, E., & Nayef, N. (2022). An efficient, font independent word and character segmentation algorithm for printed arabic text. Journal of King Saud University-Computer and Information Sciences, 34(1), 1330–1344.

[107] Rao, Y., Zhao, W., Zhu, Z., Lu, J., & Zhou, J. (2021). Global filter Networks for image classification. Advances in Neural Information Processing Systems, 34, 980–993.

[108] Rawls, S., Cao, H., Kumar, S., & Natarajan, P. (2017). Combining Convolutional Neural Networks and LSTMs for segmentation-free OCR. In Proceedings of 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 01 (pp. 155–160). Kyoto, Japan: IEEE.

[109] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015 (pp. 234–241). Cham, Switzerland: Springer International Publishing.

[110] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386–408.

[111] Sabbour, N. & Shafait, F. (2013). A segmentation free approach to arabic and Urdu OCR. Proceedings of SPIE - The International Society for Optical Engineering, 8658, 215 – 226.

[112] Sabóia, C. M. G. & Filho, P. P. R. (2022). Brazilian mercosur license plate detection and recognition using haar cascade and Tesseract OCR on synthetic imagery. In Intelligent Systems Design and Applications (pp. 849–858). Cham, Switzerland: Springer.

[113] Saha, R., Mondal, A., & Jawahar, C. V. (2019). Graphical object detection in document images. In Proceedings of International Conference on Document Analysis and Recognition (ICDAR) (pp. 51–58). Sydney, Australia: IEEE.

[114] San, L., Yatim, S., Sheriff, N., & Isrozaidi, N. (2004). Extracting contour lines from scanned topographic maps. In Proceedings of. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004. (pp. 187–192).

[115] Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., & Heer, J. (2011). Revision: Automated classification, analysis and redesign of chart images. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11 (pp. 393–402). New York, NY, USA: ACM.

[116] Schreiber, S., Agne, S., Wolf, I., Dengel, A., & Ahmed, S. (2017). DeepDeSRT: Deep learning for detection and structure recognition of tables in document images. In Proceedings of 14th IAPR international conference on document analysis and recognition (ICDAR), volume 1 (pp. 1162–1167). Kyoto, Japan: IEEE.

[117] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815–823). Boston, MA, USA: IEEE.

[118] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep Networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618–626). Venice, Italy: IEEE.

[119] Shao, M. & Futrelle, R. P. (2006). Recognition and classification of figures in PDF documents. In W. Liu & J. Lladós (Eds.), Graphics Recognition. Ten Years Review and Future Perspectives (pp. 231–242). Berlin, Heidelberg: Springer Berlin Heidelberg.

[120] Sharma, H. & Sharma, D. (2016). State of the art in Nastaleeq script recognition. International Journal of Computer Trends and Technology, 39, 40–46.

[121] Shih, F. & Chen, S.-S. (1996). Adaptive document block segmentation and classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(5), 797–802.

[122] Shin, H., Park, J., & Song, J. (2019). OCR for drawing images using bidirectional LSTM with ctc. In IEEE Student Conference on Electric Machines and Systems (SCEMS 2019) (pp. 1–4). Busan, Korea (South): IEEE.

[123] Siddiqui, S. A., Fateh, I. A., Rizvi, S. T. R., Dengel, A., & Ahmed, S. (2019). DeepTabStR: Deep learning based table structure recognition. In Proceedings of International Conference on Document Analysis and Recognition (ICDAR) (pp. 1403–1409). Sydney, Australia: University of Technology Sydney.

[124] Siegel, N., Horvitz, Z., Levin, R., Divvala, S., & Farhadi, A. (2016). FigureSeer: Parsing result-figures in research papers. In European Conference on Computer Vision (pp. 664–680). Amsterdam, The Netherlands: Springer.

[125] Simonyan, K. & Zisserman, A. (2014). Very deep Convolutional Networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[126] Singh, T. D. (2012). Bidirectional Bengali script and Meetei Mayek transliteration of web based Manipuri news corpus. In Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (pp. 181–190). Mumbai, India: The COLING 2012 Organizing Committee.

[127] Singh, T. D., Nongmeikapam, K., Ekbal, A., & Bandyopadhyay, S. (2009). Named entity recognition for manipuri using support vector machine. In Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2 (pp. 811–818). Hong Kong: City University of Hong Kong.

[128] Smith, R. (2007). An overview of the Tesseract OCR engine. In Proceedings of Ninth international conference on document analysis and recognition (ICDAR 2007), volume 2 (pp. 629–633). Parana, Brazil: IEEE.

[129] Smith, R. W. (2009). Hybrid page layout analysis via tab-stop detection. In Proceedings of 10th International Conference on Document Analysis and Recognition (pp. 241–245). Barcelona, Spain: IEEE.

[130] Su, Y., Peng, H., Huang, K., & Yang, C. (2019). Image processing technology for text recognition. In Proceedings of 2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI) (pp. 1–5). Kaohsiung, Taiwan.

[131] Sun, H.-M. (2005). Page segmentation for manhattan and non-manhattan layout documents via selective CRLA. In Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR'05) (pp. 116–120). Seoul, South Korea: IEEE.

[132] Surana, H. & Singh, A. K. (2008). A more discerning and adaptable multilingual transliteration mechanism for Indian languages. In Proceedings of the Third International Joint Conference on Natural Language Processing Hyderabad, India: AFNLP.

[133] Suzuki, S. & be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1), 32–46.

[134] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-first AAAI conference on artificial intelligence San Francisco, California, USA: AAAI.

[135] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1–9). Boston, MA, USA: IEEE.

[136] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818–2826). Las Vegas, NV, USA: IEEE.

[137] Tan, M. & Le, Q. (2019). Efficientnet: Rethinking model scaling for Convolutional Neural Networks. In Proceedings of International conference on machine learning (pp. 6105–6114). California, USA: PMLR.

[138] Tan, M. & Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In Proceedings of International Conference on Machine Learning (pp. 10096–10106).: PMLR.

[139] Tang, B., Liu, X., Lei, J., Song, M., Tao, D., Sun, S., & Dong, F. (2015). DeepChart: Combining deep Convolutional Networks and deep belief Networks in chart classification. Signal Processing, 124, 156–161.

[140] Thumthong, W., Meesud, P., & Jarupunphol, P. (2021). Automatic detection and recognition of thai vehicle license plate from CCTV images. In Proceedings of 13th International Conference on Information Technology and Electrical Engineering (ICITEE) (pp. 143–146).: IEEE.

[141] Tran, D. N., Tran, T. A., Oh, A., Kim, S. H., & Na, I. S. (2015). Table detection from document image using vertical arrangement of text blocks. International Journal of Contents, 11(4), 77–85.

[142] Tran, H. T., Nguyen, N. Q., Tran, T. A., Mai, X. T., & Nguyen, Q. T. (2022). A deep learning-based system for document layout analysis. ICMLSC 2022 (pp. 20–25). New York, NY, USA: Association for Computing Machinery.

[143] Tran, T. A., Na, I. S., & Kim, S. H. (2016). Page segmentation using minimum homogeneity algorithm and adaptive mathematical morphology. International Journal on Document Analysis and Recognition (IJDAR), 19(3), 191–209.

[144] Umer, S., Mondal, R., Pandey, H. M., & Rout, R. K. (2021). Deep features based Convolutional Neural Network model for text and non-text region segmentation from document images. Applied Soft Computing, 113, 107917.

[145] Wang, J., Li, Y., Miao, Z., Zhao, X., & Rui, Z. (2019). Multi-level metric learning Network for fine-grained classification. IEEE Access, 7, 166390–166397.

[146] Wang, S.-H., Fernandes, S., Zhu, Z., & Zhang, Y.-D. (2021a). AVNC: Attention-based VGG-style Network for COVID-19 diagnosis by CBAM. IEEE Sensors, (pp. 1–1).

[147] Wang, S.-H., Zhou, Q., Yang, M., & Zhang, Y.-D. (2021b). Advian: Alzheimer's disease VGG-inspired attention Network based on Convolutional block attention

module and multiple way data augmentation. Frontiers in Aging Neuroscience, 13, 313.

[148] White, N. (2012). Training Tesseract for ancient greek OCR. Eiiruzov, 1(28–29).

[149] Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. In Proceedings of the European conference on computer vision (ECCV) (pp. 3–19). Munich, Germany: Springer.

[150] Wu, D., Peng, Y., Zhan, Q., Ma, J., Li, G., Wei, X., Wang, X., & Zuo, J. (2021a). Text recognition technology for natural scenes. In Proceedings of 4th International Conference on Data Science and Information Technology, DSIT 2021 (pp. 212–217). New York, NY, USA: ACM.

[151] Wu, X., Hu, Z., Du, X., Yang, J., & He, L. (2021b). Document layout analysis via dynamic residual feature fusion. In Proceedings of 2021 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1–6). Taipei, Taiwan: IEEE.

[152] X. Liu, D. Klabjan, P. N. (2019). Data extraction from charts via single deep Neural Network. In arXiv preprint arXiv:1906.11906.

[153] Yang, X., Yumer, E., Asente, P., Kraley, M., Kifer, D., & Giles, C. (2017). Learning to extract semantic structure from documents using multimodal fully Convolutional Neural Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4342–4351). Honolulu, HI, USA: IEEE.

[154] Ye, P. & Doermann, D. (2013). Document image quality assessment: A brief survey. In Proceedings of 2013 12th International Conference on Document Analysis and Recognition (pp. 723–727). Washington, DC, USA: IEEE.

[155] Yokokura, N. & Watanabe, T. (1998). Layout-based approach for extracting constructive elements of bar-charts. In International workshop on graphics recognition (pp. 163–174). Berlin, Heidelberg: Springer.

[156] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation, 31(7), 1235–1270.

[157] Zhang, M., Su, H., & Wen, J. (2021). Classification of flower image based on attention mechanism and multi-loss attention Network. Computer Communications, 179, 307–317.

[158] Zhang, Y., Kong, J., Long, S., Zhu, Y., & He, F. (2022). Convolutional block attention module U-Net: a method to improve attention mechanism and U-Net for remote sensing images. Journal of Applied Remote Sensing, 16(2), 026516.

[159] Zhao, Z., Luo, Z., Li, J., Wang, K., & Shi, B. (2018). Large-scale fine-grained bird recognition based on a triplet Network and bilinear model. Applied Sciences, 8(10).

[160] Zhong, X., Tang, J., & Yepes, A. J. (2019). PubLayNet: largest dataset ever for document layout analysis. In Proceedings of International Conference on Document Analysis and Recognition (ICDAR) (pp. 1015–1022). Sydney, Australia: IEEE.

[161] Zhou, Y. & Tan, C. L. (2000a). Hough-based model for recognizing bar charts in document images. In Document Recognition and Retrieval VIII, volume 4307 (pp. 333 – 340). CA, USA: SPIE.

[162] Zhou, Y. & Tan, C. L. (2001). Learning-based scientific chart recognition. In 4th IAPR International workshop on graphics recognition, GREC, volume 7 (pp. 482–492). NY, United States: ACM.

[163] Zhou, Y. P. & Tan, C. L. (2000b). Hough technique for bar charts detection and recognition in document images. In Proceedings of 2000 International Conference on Image Processing (Cat. No. 00CH37101), volume 2 (pp. 605–608).: IEEE.

[164] Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning Transferable architectures for scalable image recognition. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 8697–8710). Los Alamitos, CA, USA: IEEE.

Publications

7.3 Publications

Journals

1. Jennil Thiyam, Sanasam Ranbir Singh, and Prabin Kumar Bora. Effect of attention and triplet loss on chart classification: a study on noisy charts and confusing chart pairs In Journal of Intelligent Information Systems September 2022, pages 1-28.

2. Jennil Thiyam, Sanasam Ranbir Singh, and Prabin Kumar Bora. Development of OCR System for Manipuri by Adaptation of OCR Systems for Languages with Comparable Scripts . In International Journal on Document Analysis and Recognition (IJDAR) *(Under review)*.

3. Jennil Thiyam, Sanasam Ranbir Singh, and Prabin Kumar Bora. Chart Classification - A Survey and Benchmarking of Different State-of-the-Arts Methods . In International Journal on Document Analysis and Recognition (IJDAR) *(Accepted)*.

4. Jennil Thiyam, Sanasam Ranbir Singh, and Prabin Kumar Bora. Integrated Document Segmentation and Region Identification-Textual, Equation and Graphical. In Multimedia System *(Under review)*.

Conferences

1. Jennil Thiyam, Sanasam Ranbir Singh, and Prabin Kumar Bora. Challenges in chart image classification: a comparative study of different deep learning methods. In In Proceedings of the 21st ACM Symposium on Document Engineering (DocEng '21), Association for Computing Machinery, New York, NY, USA, Article 29, 1–4.

2. Jennil Thiyam, Sanasam Ranbir Singh, and Prabin Kumar Bora. Chart classification: an empirical comparative study of different learning models. In Proceedings

of the Twelfth Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP '21), Association for Computing Machinery, New York, NY, USA, Article 32, 1–9.