Towards Handwriting Recognition and Search in Indic & Latin Scripts

Thesis submitted in partial fulfillment
of the requirements for the degree of

*(Master of Science in **Computer Science and Engineering** by Research*

by

SANTHOSHINI GONGIDI
2018701020
santhoshini.gongidi@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2022

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Towards Handwriting Recognition and Search in Indic & Latin Scripts" by Santhoshini Gongidi, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. C. V. Jawahar

To *Amma*

Acknowledgments

I would like to express my deepest gratitude to professor C.V. Jawahar for his guidance and support. It was wonderful learning experience to be a part of CVIT lab. I would also like to thank Kartik, Praveen, Ajoy, Minesh and other lab mates for providing feedback and guiding through my thesis.

I would also like to extend my gratitude to the CVIT staff and administrative team - Mahender, Rohitha, Aradhana, Ram, Varun and folks at CVIT annotation lab for helping me with the dataset collection process and for making all the other administrative tasks hassle-free.

I'm especially grateful for my friends who provided immense support during tough times. I would like to thank Chris, Shyam, Ashutosh, Riya, Chandu, and Prachi. Finally I would like to thank my family for their constant support and encouragement.

Abstract

ML-powered document image analysis approaches can enable intelligent solutions in bringing handwritten information into the digital world. Two major components of handwriting understanding include handwritten text recognition(HTR) and handwritten search. The former task enables the conversion of handwritten text to digital format, whereas the latter task provides easy access to the handwritten information scattered across books, archives, manuscripts and so on. We primarily focus on both these problem statements in this thesis.

Handwritten document image analysis for Indic scripts is still in its nascent stage compared to Latin scripts. For example, many commercial applications and open-source demonstrations are available for Latin scripts and there are hardly such known applications for Indic scripts. It is challenging to develop solutions for Indic scripts due to (i) variety of scripts within the Indic subcontinent, (ii) lack of huge annotated datasets and challenges in collecting data for multiple scripts, and (iii) inherent challenges in Indic scripts like inflections, joining multiple glyphs to form an akshara(equivalent to a character in Latin scripts). While challenging, it is also crucial to develop HTR and handwritten search approaches for Indic scripts. Therefore, this thesis is majorly focused on discussing approaches for Indic HTR and Indic handwritten search.

In the last two decades, large digitization projects converted paper documents and ancient historical manuscripts into digital forms. However, they remain often inaccessible due to the unavailability of robust HTR solutions. Recognizing handwritten text is fundamental to any modern document analysis system. In recent years, efforts toward developing text recognition systems have advanced due to the success of deep neural networks and the availability of annotated datasets. This is especially true for Latin scripts. In this thesis, we discuss the standard text recognition pipeline that comprises various neural network modules. Then, we present a simple and effective way to improve the text recognition pipeline and training approach. We report the improvement from our approach on four benchmark datasets in Latin and Indic scripts.

The existing state-of-the-art approaches for Latin HTR and Latin handwritten search are highly data-driven. Due to the lack of availability of large-scale data, developing Indic HTR and Indic handwritten search is challenging. Therefore, we release a collective Indic handwritten dataset with text images from

majorly spoken 10 Indic scripts. We establish a high baseline for text recognition in prominent Indic scripts. Our recognition scheme follows the contemporary design principles from other recognition literature, and yields competitive results on English. We also explore the utility of pre-training for Indic HTRs. We hope our efforts will catalyze research and fuel applications related to handwritten document understanding in Indic scripts.

Finally, we investigate the problem of handwritten search and retrieval for unlabeled collections. Handwritten search pipelines are needed in online platforms like E-libraries and digital archives. Such pipelines can efficiently search through handwritten collections and present relevant results, much like Google Search. With its ease of access and time-saving capability, the handwritten search application can prove to be valuable to many communities that study such historical documents. In this thesis, we present one such pipeline for handwritten search that performs retrieval on new and unseen collections. The proposed retrieval is not fine-tuned for specific writing styles or unknown vocabulary in the new collection. Therefore, it can be applied to new unlabeled collections.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Handwriting & Challenges

Intelligent systems that understand handwritten text add tremendous value to multiple applications in today's world. To name a few, applications like digital libraries, automatic assessment tools, automatic form processing and personalized note-taking tools stand to gain a lot from handwriting understandable intelligent solutions. A highly advanced tool for handwriting can tackle problems like handwritten text recognition(HTR), keyword spotting(KWS), content matching, handwriting imitation, layout understanding, and document summarization. These solutions can enrich the user experience by enabling easy information access. Manuscripts and other handwritten documents converted to digital text formats via HTR can be distributed as e-books. Such digital books provide access to a vast amount of information hidden away in historical documents, manuscripts and handwritten forms. In digital libraries, enabling the search feature for handwritten documents saves valuable time for many user groups like historians, language researchers, and literature researchers. KWS and HTR approaches can be utilized to make handwritten documents search-friendly. Intelligent assessment tools for handwriting depend on spotting and content matching approaches to evaluate and score handwritten submissions. Such assessment tools can check for plagiarism across submissions and summarize highlights to simplify the task of scoring. Usage of note-taking apps on tablets is currently on the rise due to easy storage and retrieval capabilities. Note-taking apps also utilize HTR tools to convert handwriting to digital text formats. Handwriting imitation finds its place in applications that make personalized notes for the user through speech input. The above usages make it exciting to build intelligent systems for handwritten data.

While the applications are exciting, the domain of handwriting comes with its challenges:

- The nature of handwriting is challenging compared to other text modalities like printed text and natural scene text. Every individual has their own distinct style of writing. An individual's style of writing also varies from document to document and situation to situation. This results in innumerable possibilities for writing a particular word.

- Along with these intrinsic challenges, the source of the documents also increases the difficulty level. For example, when dealing with historical documents, systems must adapt to text degradation, ink-bleed, and various paper textures. In modern settings, the word images can be captured in unconstrained settings like low-resolution images with poor visibility, low visibility of ink, and illegible writing. Therefore, building a good generalized system for handwriting is tricky and difficult.

- At least $3,866$ writing systems are used in different parts of the world. Handwriting understanding systems require the knowledge of a particular script to perform the given task. Each script has its own characteristics: left-to-right writing style, vertical writing style, cursive characters, conjunctive characters and so on. Given that data-driven models are the driving force behind the advancement of intelligent systems like HTR, datasets per writing system are imperative to enable progress for the overall domain of intelligible handwriting systems.

1.2 Problem Context

In today's world, handwriting exists in two formats: offline and online. Offline format refers to the natural way of writing on paper and online format refers to writing on electronic gadgets like tablets and phones. Intelligible handwriting solutions are needed for both offline and online formats. In offline format, handwritten documents are essentially images and require image processing to extract necessary information. Handwriting in online format comes with additional stroke information for processing. Due to this difference, different solutions are needed to tackle the challenges in these formats. This thesis focuses on solving problems associated with offline handwriting format. Historical collections, manuscripts, judicial records, handwritten forms, handwritten assignments and modern journals are examples of offline handwritten records. The solutions developed in this thesis can be applied to these different document categories.

Within the context of offline handwriting systems, we work on improving the recognition accuracy and propose an end-to-end pipeline for handwritten search. In both these domains, deep learning based approaches are hugely successful. These approaches enable learning highly complex functions to make the handwritten data machine understandable. The performance demonstrated by deep learning based approaches is astonishing in various domains and is not limited to handwriting. Today, we see its application in safety-first domains like healthcare and autonomous driving. Therefore, we are motivated to use deep learning as a medium to solve the problem of handwriting recognition and handwritten search. State-of-the-art text recognizers use a hybrid CRNN(convolutional recurrent neural network) architecture with rectification networks like spatial transformation network(STN). We improve upon this architecture by tweaking the training setup to include multi-head training. For the handwritten search problem, we learn powerful word image representation using a convolutional neural network(CNN) coupled with robust indexing and approximate nearest neighbour search. The proposed solution works in a zero-shot setting. In simpler words, we can demonstrate impressive search results on novel handwritten collections that are unlabeled and unseen by our training pipelines.

Research in Indic scripts for handwriting is still in its infancy. The reasons for this are multi-fold. The primary reason for this remains to be a lack of annotated data availability in multiple Indic scripts written over the subcontinent and the associated challenges in gathering handwritten data from various Indic scripts. On the other hand, the modern HTR approaches are largely data-dependent. Additionally, inherent complexities in Indic scripts due to the *abugida* writing system makes it challenging to build robust and error-free Indic HTR. For example, an *akshara*, character equivalent in Latin scripts, can have drastically different writing style depending on vowels and consonants used. Through this work, we make an effort to improve the scenario for Indic scripts. We introduce a collective dataset for 10 prominent Indic scripts. The datasets are huge and are comparable to the Latin dataset IAM[50] in size. In the handwriting research community, the IAM dataset is the most commonly used dataset to propose and evaluate new solutions. In this thesis, with the help of this new dataset, we study and discuss the problem of handwriting recognition and handwritten search for prominent Indic scripts.

1.3 Contribution

The major contributions of this thesis are as follows:

- Propose a novel and flexible approach to use multiple HTR networks with shared modules while training a CRNN architecture. The proposed approach boosts the performance by learning robust feature extractor networks. We report an average 18% decrease in the error rates over a baseline method on four datasets written in Latin and Indic scripts.

- Introduce a collective dataset for 10 prominent Indic scripts. As a part of this thesis, datasets have been collected in Bengali, Gujarati, Gurumukhi, Kannada, Malayalam, Odiya, Tamil and Urdu. The dataset has been released for public usage. We study and report the performance metrics on this dataset for standard HTR architectures and establish high baseline for Indic HTR. Additionally, we also explore how the weights learnt for one Indic script are transferable to another script in this chapter.

- Propose a generic embedding-based handwritten text retrieval framework. We use our Indic datasets and standard Latin datasets to learn holistic embeddings for handwritten text. We also study and report on other aspects like best indexing methods and efficient querying strategies. A real-time demonstration tool is also developed with our end-to-end pipeline. The tool currently has 4 Indic handwritten collections written in English, Malayalam and Bengali.

1.4 Thesis Outline

A brief overview of chapters and their structure is mentioned below:

- In Chapter 2, we formally define the tasks of handwriting recognition and keyword spotting. We provide an overview of seminal approaches proposed over time for the above tasks. We also discuss in detail about the publicly available datasets in Latin and Indic scripts.
- Chapter 3 is dedicated to discussing the suggested improvements in HTR architecture. We report detailed results on various Latin and Indic datasets.
- Chapter 4 presents the details about the released Indic datasets. We study different HTR architectures on Indic scripts and report the best-performing architecture. Experiments on transferability across Indic and Latin scripts is also discussed in this chapter.
- Chapter 5 discusses the proposed handwritten search pipeline. We outline in detail our approach for end-to-end search in handwritten documents. We benchmark our approach on Latin and Indic scripts.
- Finally, in chapter 6, we summarize the work done in the thesis. We also discuss a few future directions for Indic HTR and Indic handwritten search.

Chapter 2

Background

This chapter provides the necessary definitions and other granular details associated with handwriting recognition and spotting. In the first section, we formally define the task of handwriting recognition and introduce different flavors of the problem. One of goals of this thesis is to improve the performance metrics for handwriting recognition task. Therefore, we dive into some of the seminal ideas proposed in this domain. We contrast traditional feature engineering approaches to better feature learning approaches.

The second section focuses on keyword spotting. Similar to the first section, we discuss the problem definition and review influential works in this domain. Additionally, we summarize the efforts for end-to-end retrieval demonstrations and outline the lacking components in these retrieval approaches. This motivates our other significant contribution; a real-time demonstration tool for handwritten search.

The last section in this chapter sheds light on the existing datasets for handwriting that are categorized based on script and language. They are divided into three categories roughly: Latin scripts, Indic scripts and other scripts(Arabic, East-Asian scripts). We dedicate individual sections to provide an overview of the existing datasets per script. We briefly discuss the existing efforts towards HTR and KWS in these scripts.

2.1 Handwriting recognition

Generating digital transcripts for text images is an age-old problem. Natural scene images, printed text images and handwritten documents are different domains that are studied within the realm of text recognition. Although the goal is the same for these different input domains, a generic solution for all text images is infeasible. This is due to the stark differences observed across these domains. Even within handwriting recognition, different flavors of the problem exist. For example, online vs. offline recognition previously mentioned in Chapter 1, modern vs. historic HTRs, and plain text vs. mathematical/chemical expressions. In this thesis, we focus on offline and plain text in historical and modern handwriting styles. Another major sub-category is the different levels of resolution at which the text is recognized: character recognition, word-level, line-level, or page-level text recognition. Solving HTR has been attempted for all these resolutions. Complex resolutions like line-level or page-level include language semantics to improve recognition. Word n-grams, sentence structure, and grammatical rules are some examples of language semantics that are used in improving the recognition rate for line-level and paragraph text recognition models. Language models trained with common character n-grams are used in word recognizer models to improve recognition rate of novel words. In this thesis, we dedicate our efforts to improve word and line level models.

This task is formally formulated as: for an input word/line image(I), the task of the recognizer is to predict the best sequence of characters to form a prediction (W). The recognizer is generally followed by a lexicon(L) or a language model($P(W)$) to produce better predictions. Lexicon is meant to limit the predictions to a known set of words. Whereas the language model learns the language semantics to calculate the likelihood of the character sequence W. These complementary methods are especially beneficial to correct minor spelling mistakes in W. As the representative capability of the recognizer increases, a language model is learnt intrinsically. In such cases, these complementary methods are optional.

2.1.1 Traditional approaches

Earlier approaches for offline handwriting recognition task involved three steps: image preprocessing, feature extraction and an optical model. Preprocessing step includes noise reduction, normalizing contrast [54], skew [31] and slant [10, 42] corrections, converting the input image to standard resolution. These modules are used as an attempt to reduce the variability of writing style and to simply the task of the feature extraction.

Feature representations proposed for offline HTR vary from low-level features computed at pixel level to higher-level features involving shape detection. A few examples of low-level features include count of black pixels [49], pixel density [41], run-length analysis [41], size and aspect ratio [39], tracking transitions between black and white pixels [49], relative position of the center of gravity [41], projection profile based measures [52] and second-order moments [49]. Higher-level features involve gradient computation to gather information about the orientation of edges, and contours. Such features are extracted using Sobel filters, HoG. Features that identify and represent the structural elements of handwriting, such as loops [41, 28], T-crossings and other junctions [68], presence of ascenders and descenders [28] have been successfully used in the past for the HTR task. Côté et al. [12] combine these features to identify anchor points. Some works also include a feature transform module like PCA, LDA to remove highly correlated features or reduce the feature dimension.

An optical model is designed to generate a hypothesis for the given input. The extracted feature representations are forwarded to an optical model that converts extracted features into text strings. Proposed solutions for modeling handwriting can be broadly categorized into three approaches: whole-word models, part-based models and segmentation-free solutions. Whole-word models involve creating a recognition model for each word in the vocabulary. For example, word shape features extracted from images are compared against a lexicon containing shape features like profile features, presence of ascender or descender, etc. [12, 28] are a few works that use whole-word models for HTR task. Given that a single model identifies every word, building a integrated system for a large lexicon is infeasible. In part-based models, the image is divided into sub-regions corresponding to at most one character [39, 41]. Hypothesis from different parts are merged to create complete characters and post-processed further to determine full strings. The drawback of this approach is the need for good segmentation models to extract meaningful sub-regions from word images.

In the segmentation free approach, recognition is performed without explicit segmentation of the image into character images. The optical models used in the approaches are mainly Hidden Markov Models(HMM) [37] and neural networks [25]. HMMs can transform a sequence of feature vectors into a sequence of characters. This led to the proposal of the sliding window approach, where a small analysis window, only a few pixels wide, is slid along the text-line image. An in-depth discussion on HMMs for handwriting recognition can be found in [56]. Other segmentation free approach involve decoding the two-dimensional image into a sequence of characters. For example, Chevalier et al. [11], Lemaitre et al. [48] model a two-dimensional HMM with Gaussian modeling of spectral features.

2.1.2 Feature learning approaches

In the previous section, we briefly discussed HTR approaches that involve feature engineering to develop high-level semantic features to build an optical model for recognition. This section focuses on automated approaches that employ neural architectures to accomplish feature extraction. Neural architectures are optimized with objective functions to extract feature representations crucial for text recognition. Deep learning based approaches for HTR report significant improvement over the traditional approaches.

Connectionist Temporal Classification(CTC) layer, introduced by Graves et al. [24] for phonetic labeling task on a speech corpus, enabled much of the progress for the text recognition task. CTC layer in handwriting recognition allows the generation of a final label sequence without the need for pre-segmentation of characters in a word/line image or post-processing to generate a meaningful label sequence. The end-to-end HTR network consists of a feature extractor and the CTC layer that is trained with gradient back-propagation algorithms. Various feature extractors have been proposed for handwriting recognition task, like multi-layer perceptron, Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN). Multi-dimensional Long Short Term Memory(MD-LSTM) network, a variant of RNN, has multiple LSTM layers running along both the image dimensions to capture 2D contextual information. MD-LSTM have been successfully applied to HTR task due to the sequential nature of handwriting. This approach remained the state-of-the-art approach [55, 77] for HTR until CNN were used as feature extractors. CNN combined with RNN proved to be excellent feature extractors for text recognition [69], popularly referred to as CRNN architecture. A few architectural variations proposed for CRNN includes CNN-BLSTM [58], GCRNN(Gated CRNN) [8] and other hybrid architectures.

Recently attention modeling approaches are being widely adopted for text recognition. Architectures using attention modeling follow an encoder-decoder architecture, where the encoder generates a fixed length representation from the input sequence and the decoder transforms the encoded representation to output a target label sequence. Attention mechanism when used in the decoder, focuses on the relevant fragments of the encoded representation to generate a semantic sequence. Attention-based HTR approaches are especially meaningful for line-level or paragraph-level text recognition as attention allows to capture context in long input sequences [51, 38].

In this thesis, we focus on word-level recognition and observe that the above CTC framework outperforms the attention-based encoder-decoder architectures. Experimental results for this claim are discussed in the upcoming chapter. Therefore, we adopt the CRNN with CTC framework to train robust handwriting recognizers.

2.2 Search & Retrieval from handwritten documents

E-libraries and digital archives serve the handwritten collections to the user through scanned images. Searching for text content from such images is time-taking and cumbersome. Users have to read through the entire collection to find the necessary content. Automatic retrieval pipelines are needed in such online platforms to reduce the burden on the user. Such pipelines can efficiently search through handwritten collections and present relevant results, much like Google Search. With its ease of access and time-saving capability, the handwritten search application can prove to be valuable to many communities that study such historical documents.

A naive way to approach handwritten search is to utilize the transcriptions from HTRs. It is more straightforward to search from transcript text files than to search from document images. HTR-based approaches also allow advanced search operations like querying in natural language and summarization. In the past, recognition-based methods were used to solve this problem. However, even the best HTR recognizers generate unreliable transcriptions due to variations observed across handwriting, document degradation, and text written in ancient calligraphic fonts. Additionally, new and unseen words are difficult to predict for an HTR. Unreliable and erroneous HTR predictions drastically impact the precision of search operations.

Later, the concept of keyword spotting was introduced for retrieval applications. KWS is defined as the task of identifying the occurrences of a given query from a data collection. Formally, KWS for the handwriting domain is defined as detecting query words($Q_{[w1,w2]}$) in handwritten document image collections ($C_{[d1,d2]}$) without involving a traditional OCR step. Spotting query strings across images involves textual search. It is the most common method used for search. Alternatively, an exemplar search involves using relevant images as input queries. It is especially beneficial in the handwriting domain. Exemplar search simplifies the search process and has additional applications like searching for unknown or unidentified symbols. These search modes are referred to as *Query-by-String(QbS)* and *Query-by-example(QbE)*. Current spotting approaches for search and retrieval are very promising for both textual and exemplar searches.

2.2.1 Word Image Representation

2.2.1.1 Handcrafted Feature Representation

Word images are processed to extract holistic features using image processing techniques. Various types of global and local features have been proposed for KWS since the idea of KWS was introduced in [59, 60]. This seminal work adopts profile-based features like projection profile, upper and lower profile, and feature sets obtained from Gaussian smoothing and its derivatives. Since then, profile-based features, shape and structural features, contour matching, a bag of features with keypoint descriptors, and many more approaches have been presented to solve KWS for Latin and non-Latin scripts. These various features for word image representation are surveyed and detailed in [1].

A chosen set of features are computed and combined to create a suitable representation to aid query matching process. These representations can be of variable lengths or fixed length vectors. Other representation include probabilistic representation, graph-based features, Word Shape Coding(WSC), Fischer vectors and Bag-of-Visual-Words(BoVW). We request the readers to refer [20] for better understanding about feature representations.

2.2.1.2 Data-driven Representation Learning

Recently, deep-learning based feature learning methods report incredible performance for KWS task on benchmark datasets like IAM [50], and GW [18] datasets. Feature learning methods are highly data-driven, where a model is trained to learn a holistic representation for word images and their corresponding text strings.

A key idea in representation learning for word images is attribute engineering for text strings. An attribute is an intermediary mapping between a word image and a text string. We discuss two such popular attributes proposed in the domain of KWS, PHOC [72] and DCToW [79]. These attributes are described figuratively in Fig. 2.1. Pyramidal Histogram of Characters(PHOC) attribute for a word is a pyramidal binary histogram of the occurrences of the characters and n-grams in multiple splits of that word. PHOC captures the spatial presence of specific characters/n-grams in a word. Whereas, Direct Cosine Transform of Words(DCToW) is a low frequency, distributed representation of a word. Direct Cosine Transform applied on the one-hot encoded matrix of the word is cropped, retaining only low-frequency components for every row in the one-hot matrix. This flattened low-frequency vector forms DCToW attribute for the word.

Some of the works that use attributes to propose spotting approaches are: attribute learning in PHOCNet [72], verbatim and semantic feature learning in triplet CNN [79], and common representation learning in HWNet [45]. PHOCNet is a deep CNN that predicts PHOC representation for a given word image. Spotting is achieved by matching PHOC vectors of word images and text strings. In [79], a triplet CNN is trained to generate an image representation space. A two-layer embedding network then converts the image embedding to a target word embedding space, like PHOC or DCToW attributes. Alternately, HWNet approaches the problem of spotting by learning a common embedding space for

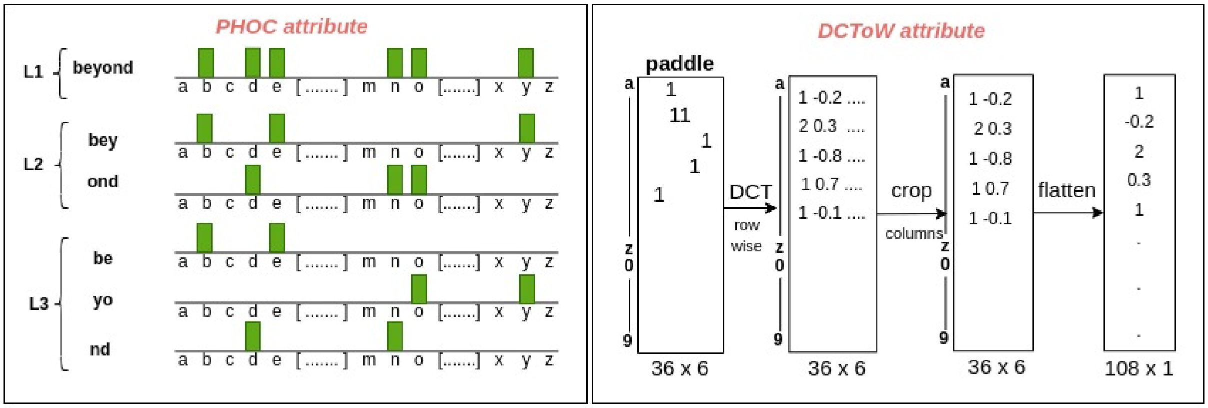


Fig. 2.1: Commonly used word attributes for text string representation: PHOC [72] and DCToW [79].

word attributes and word images. The performance metric Mean Average Precision(MAP) reported on IAM dataset for these methods is above 0.82 and 0.75 for QbE and QbS settings respectively. Due to the incredible performance of these embeddings, we are motivated to explore embedding based spotting approaches for the handwritten search and retrieval pipeline proposed in this thesis.

The works mentioned above require segmented word images to spot queries in a document collection. These methods fall into segmentation-based approaches for spotting. However, a segmentation module is necessary to build an end-to-end spotting and retrieval pipeline. Pretrained text detection networks or image processing approaches can be used as segmentation modules in a search and retrieval pipeline. Segmentation-free approaches also exist for KWS, where the end-to-end trainable network contains a text region proposal network coupled with the above spotting techniques. Wilkinson et al. [80] propose one such end-to-end model, Ctrl-F-Net. This work proposes a fully differentiable dense localization layer for region-based training and prediction, together with DCToW embedding-based approach for KWS. In this thesis, we prefer using a pretrained text detection network trained for generic text irrespective of language, nature of the text, and backgrounds. In an end-to-end network like Ctrl-F-Net, the text detection module is tuned for a specific manuscript, and therefore, it is infeasible to train the network for generic text detection.

2.2.2 End-to-End retrieval frameworks

An automatic retrieval system for historic manuscripts was first introduced in [62] for George Washington's letters evaluated over 987 document images. A statistical relevance based language model is proposed in this work. This language model is learnt from the word image and text string pairs available in the annotated set of 100 pages from the George Washington collection. Word images are processed to compute holistic shape features to create a feature vocabulary. For indexing, word images in the test set are assigned probabilities for every word in the annotation vocabulary. Given a text query, retrieval can be done with these probabilistic annotations in a language model based approach using query-likelihood ranking.

Recently, another concept referred to as probabilistic indexing(PrIx) has been introduced. In this method, pixel-level posterior probability $P(w|I, i, j)$ for a word(w) given a text image(I) are computed using a contextual n-gram based word classifier. The probabilities are computed for query words from vocabulary V for all the images present in a document collection. These computed probabilities, along with position information are indexed. For a user query, this index is searched to retrieve relevant results. Using PrIx approach, multiple large-scale Latin collections have been made search-friendly. Bentham papers [74, 53], Chancery manuscripts [7], Carabela manuscripts [76], and Spanish TSO collection [74] are a few sample manuscripts that have online demonstrators[1]. PrIx demonstrators require a substantial number of transcribed pages to make a collection indexable. For Carabela manuscripts and Bentham papers, 558 and 1213 transcribed pages are used to train the n-gram based word classifiers.

A case study on Swedish court records, presented in [80], discusses a segmentation-free word spotting approach. This work proposes a QbS end-to-end trainable word spotting model. To handle the segmentation component in the end-to-end pipeline, a region proposal network(RPN) is trained. The proposed text regions are processed to compute PHOC and DCToW embeddings for query matching. 11 transcribed pages from the Swedish court records collection are needed to fine-tune the word spotting model.

The methods above rely on transcribed pages from the collection to make it retrievable. The number of pages to be transcribed varies for different methods. The transcription costs are infeasible as millions of massive collections written in different scripts exist in digital archives. Another drawback of the above methods is the additional retraining costs to fine-tune the model for newer collections. These existing demonstrations are developed explicitly for Latin scripts. No such retrieval demonstrations are available for non-Latin scripts.

Our proposed pipeline performs retrieval on unseen collections without any need for fine-tuning. At the same time, our collections are equally complex compared to the above-mentioned collections. As the pipeline is not fine-tuned for specific writing styles or vocabulary of a new collection, our work comes under the category of zero-shot learning.

2.3 Handwriting Datasets

Solutions developed for intelligent handwriting systems are heavily data-driven today. Most of them use machine learning methods and learn from annotated examples. This demands for script/language-specific collection of examples. For machine learning methods to be effective, the annotated datasets should be substantial, voluminous in size, and diverse in nature. For handwriting systems, this implies that the dataset should have (i) many writers, (ii) extensive vocabulary, and (iii) a huge number of samples. In the past, public datasets such as IAM [50] and George Washington [18] have catalyzed the

[1] http://prhlt-carabela.prhlt.upv.es/PrIxDemos/

Table 2.1: Publicly available handwritten datasets for Latin scripts. The table lists both modern and historic handwritten datasets. The GT level mentioned in the table is the finest level of annotation available per dataset.

Dataset	Script	GT Level	#Writers	#Instances			#Lexicon
				Pages	Lines	Words	
READ'18[71]	German & Italian	Line	5	27	16984	98239	10967
READ'17[66]*	French, German & Italian	Line	-	115	2959	30049	9272
Bentham[67]*	English	Line	-	471	12487	113854	11800
CVL[40]	German & English	Word	311	1604	13743	100174	508
IAM[50]	English	Word	657	1539	13353	115320	10841
GNHK[47]	English	Word	-	687	39026	9363	12341
Parzival[19]	German	Word	3	47	4477	11743	4934
GW[18]	English	Word	2	20	656	4894	1471
RIMES[26]	French	Character	1300	12723	12111	51916	1636

* Additional training subsets available with just page level transcriptions.

research in handwriting recognition and retrieval, especially for Latin scripts. Even today, efforts are being made to introduce better handwritten datasets in Latin scripts and extend the efforts to collect datasets in other scripts. Below, we provide a brief overview of publicly available datasets written in Latin and Indic scripts below.

We also briefly discuss synthetic datasets for handwritten data. Synthetic datasets are automatically rendered images using font files that imitate handwriting styles. Works [45, 14] show that synthetic data highly improves the accuracy of handwritten text recognition and retrieval. In this thesis, we worked with datasets written in Latin script, Indic scripts and synthetic handwritten datasets.

2.3.1 Latin scripts

Publicly available Latin datasets, listed in Table 2.1, are the enablers for the progress seen in the domain of handwritten text recognition and search. In the table, we list the dataset along with the dataset size, lexicon size, writer variability and script variability to show the diverseness of Latin handwritten datasets.

Datasets for Latin scripts such as IAM [50] and Bentham collection [65] have over 100K running words with large lexicons. The IAM dataset is one of the most commonly used datasets in HTR for performance evaluation even today. This dataset contains $115,320$ word instances written by 657 writers and has a lexicon size of $10,841$. IAM provides annotations at word, line, and page levels. IAM is the most versatile dataset available for handwritten data covering large vocabulary with a huge number of samples from many writers. As the performance on IAM dataset has started to saturate, more challenging datasets have started to surface. The historical datasets such as the Bentham collection [65] and the READ dataset [66, 71] are associated with line-level and page-level transcriptions. Annotations at the line, word, and document level provide opportunities for developing methods that use language models

Fig. 2.2: Examples from publicly available datasets in Latin and Indic scripts. The dataset titles are listed on the left edge of the image.

and higher order cues. Parzival dataset [19], containing historical manuscripts from the medieval period, is generally used for historical document layout analysis. On the other hand, CVL dataset [40] has the smallest vocabulary and therefore, it is not suitable for HTR benchmarking. CVL is used for studying and reporting benchmarks for keyword spotting task. GNHK dataset is a recently released dataset that captures handwriting styles seen in the world. It comprises unconstrained camera-captured images from four different geographical regions. Fig. 2.2 shows a few examples from the above mentioned datasets. In this thesis, we use IAM, GW and RIMES to report the performance on Latin script HTR.

2.3.2 Indic scripts

The lack of large handwritten datasets remains a significant hurdle for developing robust solutions to Indic HTR. Table 4.1 presents the list of publicly available datasets in Indian languages.

Most Indic handwritten datasets are smaller than the IAM in the number of word instances, writing styles, or lexicon size. For example, the lexicon used to build the CMATER2.1 [6] and the TAMIL-DB datasets [73] consist of city names only. The CENPARMI-U dataset provides only 57 different financial terms in the entire dataset. The small and restrictive nature of the lexicon for these datasets limits the utility while building a generic HTR to recognize text from a large corpus of words. ROYDB [63] and LAW datasets [35] use a larger lexicon. However, both the datasets are small and possibly insufficient to capture the natural variability in handwriting. The PBOK [2] dataset provides the page-level transcriptions for 558 text pages written in three Indic scripts. Page-level text recognizers require an excessive amount of data compared to word-level recognizers. On the IAM and READ datasets, the state-of-the-art full-page text recognizer [82] has high error rates due to additional challenges like extremely skewed

Table 2.2: Publicly available handwritten datasets for Indic scripts. All the datasets provide word-level transcriptions, except for the PBOK dataset.

Dataset	Script	#Writers	#Word Instances	#Lexicon
PBOK [2]	Bengali	199	21K	925
	Kannada	57	29K	889
	Odia	140	27K	1040
ROYDB [63]	Bengali	60	17K	525
	Devanagari	60	16K	1030
CMATERDB2.1 [6]	Bengali	300	18K	120
CENPARMI-U [64]	Urdu	51	19K	57
LAW [35]	Devanagari	10	27K	220
TAMIL-DB [73]	Tamil	50	25K	265
IIIT-HW-DEV [15]	Devanagari	12	95K	11,030
IIIT-HW-TELUGU [16]	Telugu	11	120K	12,945

text, overlapping lines and inconsistent gaps between lines. Note that the IAM dataset is $3\times$ bigger than PBOK dataset. Therefore, solving Indic HTR for full-page text recognition requires a major effort and is beyond the scope of this work.

IIIT-HW-DEV [15] and IIIT-HW-TELUGU [16] are possibly the only Indic datasets comparable to the IAM dataset at the word level. These datasets have around 100K word instances, each with a lexicon size of over 10K unique words. In this work, we complement this effort and extend similar datasets in many other languages to cover ten prominent Indic scripts. We introduce a unified database for handwritten datasets comparable to the IAM dataset in size and diversity. This dataset for Indic scripts is referred to as the IIIT-INDIC-HW-WORDS dataset. Indian languages are still in their infancy as far as HTRs are concerned, and we limit our attention to creating word-level annotations in this work. We describe the dataset in Chapter 4

2.3.3 Synthetic Datasets

Dataset creation for handwritten data is an expensive and time-consuming process. It involves multiple steps, from planning and collection to annotation and validation. As discussed in the above sections, today's approaches for both HTR and KWS rely on large datasets with multiple attributes like extensive vocabulary and multiple writers. Introducing synthetic datasets into the experimentation setups reduces the need for extensive datasets. Such datasets can be generated artificially using synthetic rendering mechanisms for the handwriting domain.

Here, we discuss datasets and tools introduced to generate synthetic datasets for handwriting data. IIIT-HWS dataset contains over 9 million synthetic word images generated from 750 unique font styles. The dataset is generated by sampling from 90K colossal vocabulary. Fig. 2.3 shows a few sampled images from the synthetic dataset. The dataset is generated using ImageMagick library. Another tool

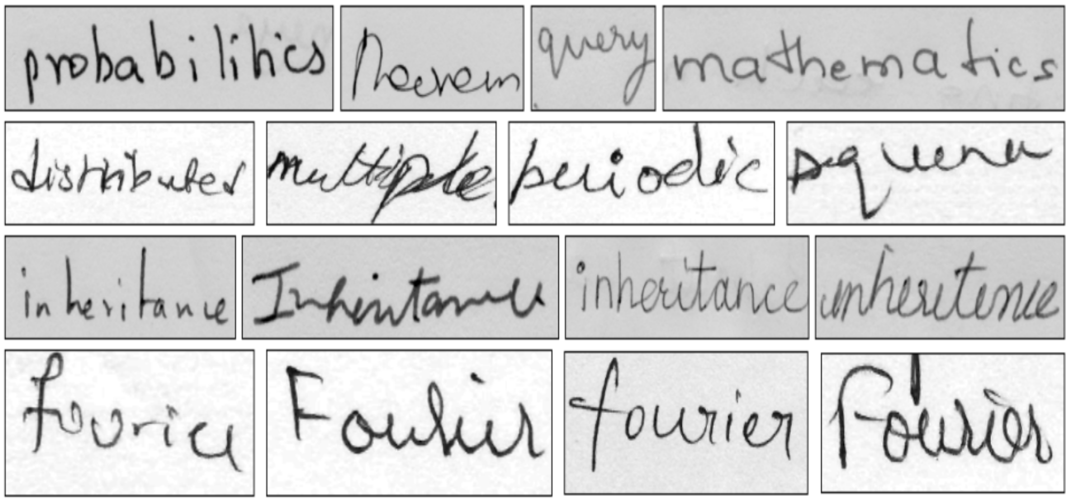


Fig. 2.3: Synthetic word images from in the IIIT-HWS dataset. The first two rows show the same word rendered by different fonts. The last two rows show different words being rendered by different fonts.

available for Latin data generation is TRDG[2]. It is an open-source library to render text lines scrapped from Wikipedia as images.

[2]https://github.com/Belval/TextRecognitionDataGenerator

Chapter 3

Handwritten Text Recognition

3.1 Introduction

The problem of end-to-end text recognition is generally broken down into various sub-tasks like layout analysis, localization and recognition. In this work, we focus only on recognizing the text from word in an unconstrained setting. Unconstrained offline handwritten text recognition (HTR) is the task of identifying the written characters in an image without using any external dictionaries for a language. Previous works in text recognition[69, 75, 4, 8, 58] show that deep neural networks are very good at solving this problem because of their generalization ability and representation power. Convolutional Recurrent Neural Network(CRNN) proposed by Shi et al.[69] has gained much attention in the domain of text recognition due to its performance, and its capability to handle variable width images. It also does not use fully connected layers in the network making it a very compact model. After this work, multiple other architectures with alternate objective functions have been proposed for HTR.

Kartik et al.[17] extend the above CRNN architecture and propose a similar architecture with a STN head and discuss different strategies to achieve error rates less than 5% on IAM dataset. In this thesis, we consider this architecture as a baseline method to study alternate modules and propose novel improvements to the architecture. In the later sections, we discuss the architecture and its different modules in detail. We propose a novel approach to boost the performance of the baseline architecture. We evaluate our HTR architecture on publicly available datasets in Latin and Indic scripts.

3.2 HTR Architecture Pipeline

Several deep neural network architectures have been proposed for the recognition of scene text, printed text, and handwritten content. Baek et al. [4] propose a four-stage text recognition framework derived from existing scene text recognition architectures. The recognition flow at different stages of the framework is demonstrated in Fig. 4.5. This framework can also be extended to the HTR task. In this section, we discuss and study the different stages of the pipeline.

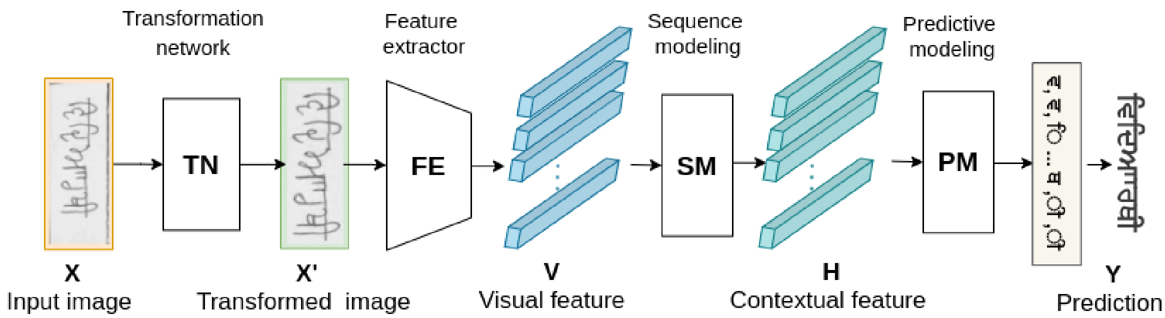


Fig. 3.1: Generic pipeline for text recognition. Here, we demonstrate the flow of a sample image **X** through the pipeline to generate a text prediction **Y**. The sample shown here is written in Gurumukhi script.

3.2.1 Transformation Network (TN)

Diverse styles observed in handwriting data are a significant challenge for HTR. A transformation block learns to apply input-specific geometric transformations such that the end goal of text recognition is simplified. This module simplifies the input variations for feature extractor and sequence modeling architectures. Architectures like spatial transformer network (STN) [33] and its variants are commonly used to rectify the input images. In this work, we mainly focus on STN[33] architecture and tweak the existing block to introduce alternate TN blocks.

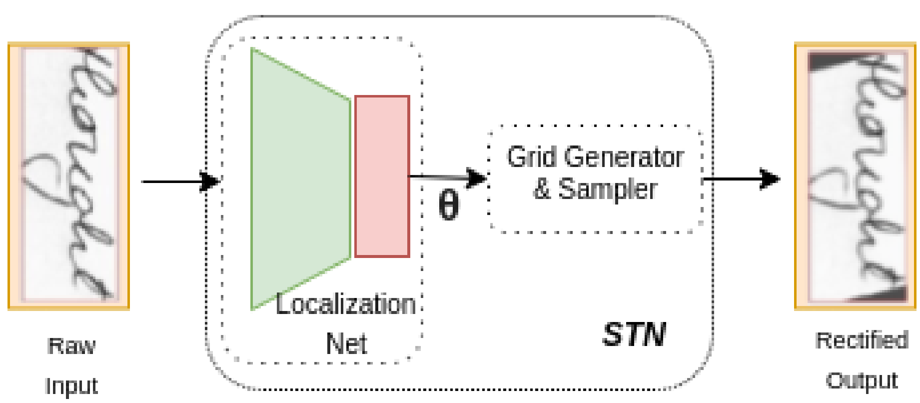


Fig. 3.2: Block diagram representation of STN architecture. Here, the raw word image, containing the string *thought*, is transformed by STN to generate rectified output word image. The STN transformation applied here is a vertical shearing transformation.

STN is a trainable network consisting of a localization network, a grid generator and a sampler. Fig. 3.2 shows a pictorial representation of STN block with its components. STN uses the input image to generate a transformation defined by parameters θ. Image transformations, in general, are defined by a set of parameters and the localization network in STN module is designed to learn to generate the transformation parameters θ. The learnt parameters θ are used by the grid generator to produce an output image grid. Sampler module samples the raw input image according to the generated image grid

Table 3.1: Architecture of localization net in STN. Here, H and W refer to height and width, respectively. And other notations c represents number of channels, k is the kernel size, s is the stride, p is the padding and h is the hidden layer size.

Layer	Feature map ($H \times W$)	Configuration	Activation
Input	32×64	resized grayscale image	$-$
Conv0	32×64	$c:16 \quad k:3\times 3$	$ReLu$
Block0	32×64	$\begin{bmatrix} c:16 \quad k:3\times 3 \\ c:16 \quad k:3\times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} ReLu \\ - \end{bmatrix}$
AvgPool	4×8	$k:8\times 8 \qquad s:8\times 0$	$-$
FCBlock	1×6	$\begin{bmatrix} h:30 \\ h:6 \end{bmatrix}$	$-$

to produce a rectified output image. In case of affine transformation, 6 parameters are learnt to define an affine transformation. Affine STN enables the HTR module to become invariant to rotation, scale and skew changes in word images.

STN's trainable parameters, the weights of localization net, are fine-tuned implicitly with the objective loss function between the predicted label sequence and target label sequence. The architecture stack used for localization net in STN is shown in Table 3.1. In the stack $Conv0$, and $Block0$ uses convolution layers. Each convolution layer is followed by batch normalization and activation operation. Whereas the $FCBlock$ comprises 2 fully-connected layers. Residual skip connections are used in *Block0* to overcome the vanishing gradient problem in STN.

3.2.2 Feature Extractor (FE)

The rectified image from the TN module is forwarded to the FE module to extract the visual feature sequence. The output visual sequence is defined as $V = v_1, v_2, ...v_W$, where W is the width of the image. The sequence V is used to predict characters for each vector v_i by the later modules in the text recognition pipeline.

Convolutional neural networks(CNN) are the most popular feature extractors. The feature map generated by a CNN is converted into a visual sequence vector V with a map-to-sequence operation. This operation transforms the feature map of size $c \times h \times w$ into w feature vectors where each vector v_i has size $c \times h$. Each vector v_i corresponds to a receptive field along the width of the word image. Therefore, a vector V_i represents the character present in its corresponding receptive field. Fig. 3.3 pictorially depicts the functioning of a feature extractor module. Words are recognized sequentially by predicting the characters at every step of the visual sequence V.

For FE module, we adopt RESNET architecture introduced by He et al. [30]. Residual connections across layers ease the training of deep CNNs. The network configurations are tweaked such that the gen-

Table 3.2: Architecture of HW-RESNET. For an input image of size 96×256, HW-RESNET generates 512 output feature maps of size 1×32.

Layer	Feature map $(H \times W)$	Configuration	Activation
Conv0	96×256	$c : 64 \quad k : 3 \times 3$	$ReLu$
MaxPool0	48×128	$k : 2 \times 2 \quad s : 2 \times 2$	–
Block0	48×128	$\begin{bmatrix} c : 64 & k : 3 \times 3 \\ c : 64 & k : 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} ReLu \\ - \end{bmatrix}$
MaxPool1	24×64	$k : 2 \times 2 \quad s : 2 \times 2$	–
Block1	24×64	$\begin{bmatrix} c : 128 & k : 3 \times 3 \\ c : 128 & k : 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} ReLu \\ - \end{bmatrix}$
MaxPool2	12×32	$k : 2 \times 2 \quad s : 2 \times 2$	–
Block2	12×32	$\begin{bmatrix} c : 256 & k : 3 \times 3 \\ c : 256 & k : 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} ReLu \\ - \end{bmatrix}$
MaxPool3	6×33	$k : 2 \times 2 \quad s : 2 \times 2$	–
Block3	6×33	$\begin{bmatrix} c : 512 & k : 3 \times 3 \\ c : 512 & k : 3 \times 3 \end{bmatrix} \times 2$	$\begin{bmatrix} ReLu \\ - \end{bmatrix}$
MaxPool4	3×34	$k : 2 \times 2$ $s : 2 \times 1 \quad p : 0 \times 1$	–
Conv1	1×32	$c : 512 \quad k : 3 \times 3$ $s : 1 \times 1 \quad p : 0 \times 0$	$ReLu$

erated feature maps have a larger width. To correctly recognize longer words, longer feature sequences need to generated. The adjusted configuration generates longer feature sequences. The network configuration of the tweaked CNN is shown in Table 3.2. Batch normalization is done after each convolution operation followed by activation operation. We refer to this architecture as HW–RESNET.

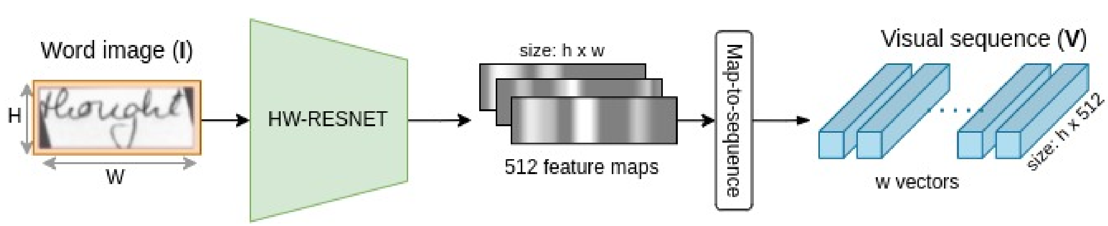


Fig. 3.3: Feature Extractor module in HTR pipeline. HW-RESNET is a CNN with residual connections. The computed feature maps are restructured to generate a visual sequence V that aligns with characters seen along the width of the word image.

3.2.3 Sequence Modeling (SM)

The computed visual sequence V lacks contextual information necessary to recognize characters across the sequence. Contextual cues embedded into a feature sequence reduce ambiguity and make it easier to predict the final word sequence. For example, when observing their previous and next characters, differentiating between characters 'a' and 'o' is easier. Therefore, a stack of recurrent neural networks is introduced to capture the contextual information and compute a contextual feature $H = \{h_i\}, i = 1..., W(W$ is image width). The recurrent layers predict a label distribution h_i for each step v_i in the feature sequence V. Bidirectional LSTM(BLSTM) is the preferred block to compute the new feature sequence H as it enables context modeling from both directions. Due to BLSTM success [69, 58, 17, 4], we use a two-layer BLSTM architecture with 256 hidden neurons in each layer as the SM module in our experiments.

3.2.4 Predictive Modeling (PM)

PM module is responsible for decoding a character sequence from the contextual feature H. For unconstrained handwritten text recognition task, the model needs to learn successfully to align sequence H to target labels without any prior knowledge about language, the position of characters in the feature sequence H etc. This alignment can be learnt using either a CTC-based approach or Seq2Seq approach with attention.

Connectionist Temporal Classification(CTC) layer, introduced by Graves et al. [24], is used to generate predictions in this work. CTC outputs a conditional probability $p(Y|H)$ based on the per-frame predictions. The probability is defined as follows,

$$p(Y|H) = \sum_{\pi:M(\pi)=Y} p(\pi|H)$$

where the probability of sequence π is defined as $p(\pi|H) = \prod_{t=1}^{W} y_{\pi_t}^t$ and M is a mapping function defined over the possible character set to firstly remove repeating characters and then remove CTC blank symbol. $y_{\pi_t}^t$ is the probability of generating character π_t at time step t. To compute the conditional probability $p(Y|H)$, CTC layer uses a forward-backward algorithm described in [24].

During inference time, in a lexicon-free setting, the prediction sequence with highest probability is considered as best prediction Y^* and is computed from $Y^* \approx M(\arg\max_\pi p(\pi|H))$.

3.2.5 Data augmentation

Training deep networks to learn generalized features is crucial for handwriting recognition, as handwritten data can have many variations. Augmentation and image transformations enables the network to learn invariant features for the given task. Generally, the handwritten samples are augmented using affine transformation, elastic transformation and multiscale transformation. In our experiments, grid

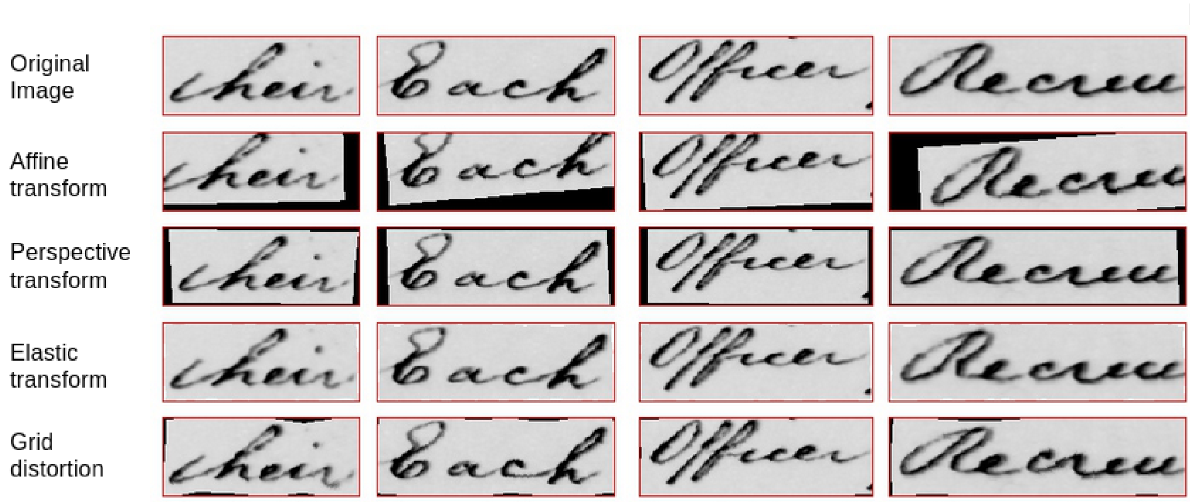


Fig. 3.4: Image transformations applied in the data augmentation pipeline. Each row shows the effect of a particular transformation on 4 word samples.

distortion[78] and perspective transformation are applied as a part of the augmentation pipeline to improve the baseline. Grid distortion is introduced as it imitates the natural distortions and variations observed in handwriting data.

Our augmentation pipeline consists of affine transform, elastic transform, perspective transform and grid distortion. The parameters defining affine transform are restricted with maximum rotation up to $(+/-)5$ degrees, maximum shear up to $(+/-)0.5$ degrees along horizontal direction and maximum translation up to 20 pixels. The distortion parameter for perspective transform is 0.1. The smoothing parameter and scale factor for Elastic distortion used is 0.06 and 0.2. Grid distortion is similar to elastic distortion, but is applied to smaller grids in the image. The grid size used in the experiments is 31 and deviation parameter which controls the displacement is set as 2.04. These parameters values are selected based on the information provided in [17, 78]. Figure 3.4 shows the augmented images obtained for these image transformations.

3.3 Proposed Method

In this section, we propose a simple yet effective way to improve the performance of the above HTR architecture. Basic training of the above HTR architecture includes optimizing the CTC objective function for the joint network of TN-FE-SM-PM blocks. To boost the performance of this architecture, we focus on extending the basic training setup. The following sections discuss the training pipeline, weight initialization and pretraining techniques required for the new training setup.

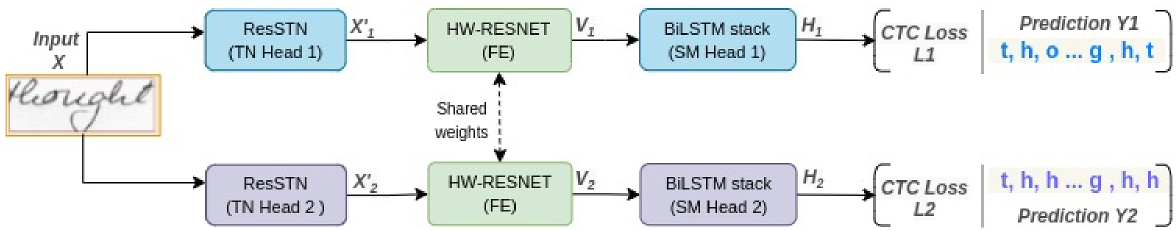


Fig. 3.5: Two-head training setup for HTR architecture. Input image rectified by two different TN blocks are fed to the shared FE network and SM heads to generate contextual features H_1 and H_2. CTC-based PM block generates the prediction Y_1 and Y_2. The HTR heads are jointly trained with the CTC loss L_1 and L_2. During test time, only one end-to-end HTR head is used for prediction.

3.3.1 Two-head Training

The proposed HTR training setup is shown in Figure 3.5. We introduce a parallel head of the HTR network during training while using a shared FE network(HW-RESNET). The TN and SM networks are independent and do not share any weights. The parallel HTR heads are trained in an end-to-end fashion with a join CTC loss from two HTR heads. The joint loss (L) is simply the sum of CTC losses: L_1 and L_2. For a training set $D = \{X_i, l_i\}$ containing word images X_i and target label sequence l_i, the end-to-end network is trained by minimizing the negative log-likelihood of the conditional probability of target sequence l_i for both the heads. The loss L_h for a HTR head h is expressed as follows,

$$L_h = -\sum_{(X_i,l_i)\in D} \log p(l_i|y_i)$$

y_i is the sequence generated from the image X_i by the HTR head h.

The advantage of this training setup is the robust training of the FE network. The visual features V_1, V_2 for an input X must be similar to predict the correct label sequence. However, the input to the FE network are the transformed images X_1', X_2' from TN heads. Given that the TE heads are initialized differently, the outputs X_1', X_2' are not similar. The FE network implicitly learns to predict similar visual features for differently rectified images while minimizing the joint CTC loss.

During test time, one of the HTR head is used for predicting label sequences. Without adding any additional parameters, the proposed training setup allows for improved training of the HTR network.

3.3.2 Pretraining and Weight Initialization

Pretraining is a common technique used while training deep networks. In our case, we pretrain the basic HTR architecture on a synthetic dataset and the real training dataset. Section 2.3 describes the synthetic data sources available for handwriting data. We use the extensive IIIT-HWS dataset to pretrain our HTR network.

To train the two-head HTR setup, the pretrained weights are used to initialize the HTR head M_1. The weights for the HTR head M_2 are randomly initialized from a normal distribution. Please note that only

TN and SM block require weight initialization. The HW-RESNET shares the pretrained weights from the M_1 HTR head.

3.4 Experiments

In this section, we report the performance metrics for the proposed approach on a few benchmark datasets. To compare the performance gain from the proposed training approach, we choose four publicly available datasets: two Latin datasets and two Indic datasets. We report the results on IAM, RIMES, IIIT-HW-DEV, IIIT-HW-TELUGU datasets. The details of these datasets are discussed in section 2.3. The networks are trained using Adadelta learning optimizer with an initial learning rate of 3×10^{-3} and a batch size 96. The baseline network is trained for 500 epochs and the proposed method is trained for 200 epochs.

The metrics used to compare different methods are described below. Word Error Rate(WER) is calculated differently for word-level datasets and line level datasets. WER for word level datasets is defined as,

$$WER = \frac{\#Correctly\ recognized\ samples}{\#Total\ samples}$$

Character Error Rate(CER) is defined as,

$$CER = \frac{\sum_{i=1}^{N} EditDistance(Gt_i, P_i)}{\sum_{i=1}^{N} Length(Gt_i)}$$

where Gt_i and P_i are the ground truth and predicted text for i^{th} sample and N is the total number of samples.

Table 3.3: Unconstrained word recognition results on HTR benchmark datasets.

Dataset	Script	Baseline Method [14]		Proposed Method	
		WER	CER	WER	CER
IAM	Latin	13.16	5.10	11.89	4.58
RIMES	Latin	7.04	2.32	6.44	1.82
IIIT-HW-DEV	Indic(Devanagiri)	11.27	4.90	17.025	3.98
IIIT-HW-TELUGU	Indic(Telugu)	23.98	4.58	18.6	3.56

To report the performance gain from the proposed training setup, we select a baseline HTR architecture and compare WER and CER metrics for the baseline method and the above approach. The baseline network is chosen such that the proposed two-head training setup is feasible. Kartik et al. [14] present a hybrid CRNN architecture with affine STN, HW-RESNET followed by a 2-layer BLSTM. This architecture is used as the baseline architecture and the two-head training pipeline is used to finetune the network further. During testing, predictions from M_1 HTR head are considered as final predictions to compute error rates. Quantitative results comparing the baseline architecture and the proposed method are shown in Table 3.3. CER improves across Latin and Indic datasets presented in the table. WER

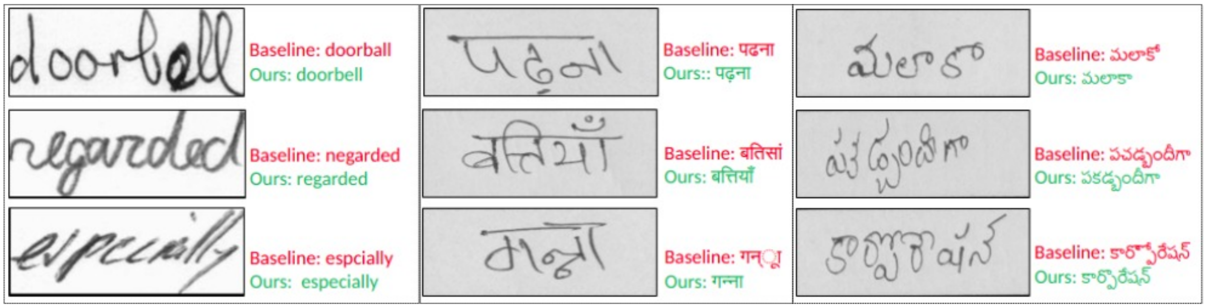


Fig. 3.6: Qualitative results comparing baseline architecture and the proposed method. For these incorrect predictions from the baseline architecture, the proposed method generates correct prediction.

also improves for all datasets, except for the IIIT-HW-DEV dataset. We also present a few qualitative examples in Fig. 3.6. The results are shown in Latin, Devanagiri, and Telugu scripts from IAM, IIIT-HW-DEV, IIIT-HW-TELUGU datasets. The proposed method is able to generate correct predictions for the misspelled prediction from the baseline method. We observe that our predictions are better at discriminating confusing character pairs like *a-o*, *v-r* and *t-l*. The same observation applies for Indic scripts too. Additionally, our HTR architecture generates fewer grammatical mistakes compared to the baseline method for Indic scripts. Third row in the Fig. 3.6 shows incorrect conjunct consonants in baseline predictions that are correctly predicted by our HTR network.

Our training approach is effective and allows the feature extractor network HW-RESNET to generate robust visual feature maps. Additionally, the tweaked STN architecture eases the training, especially for localization net parameters. The proposed training setup is flexible as it can be applied HTR architectures that follow the text recognition pipeline shown in Fig. 4.5.

3.5 Summary

We discuss the basic HTR pipeline in this chapter and propose a novel approach to finetune a given htr architecture by introducing a multi-head training approach. We successfully apply the proposed approach to four benchmark datasets in Latin and Indic scripts to show the performance gain from our method. The effectiveness of our method suggests that the current HTR architectures can be finetuned to further reduce error rates and develop robust HTR. For example, adopting approaches like constrastive learning, multi-task or joint learning allows the CNN-based feature extractor network to identify common attributes between similar images and, as a result generate better visual features critical for prediction generation.

Chapter 4

Indic Handwritten Text Recognition and Indic Datasets

In the previous chapter, we have seen an effective and hybrid CRNN pipeline for recognizing handwritten text in Latin script. In this chapter, we discuss our efforts in developing HTR for various Indian languages. The state-of-the-art approaches for handwritten recognition use deep learning based architectures that are trained on annotated handwritten datasets. Deep learning based approaches are successful because they utilize large annotated datasets to learn features from handwritten word images that can help with text recognition. The non-availability of annotated handwritten datasets is major hurdle when it comes to developing Indic HTR. As a part of this work, we have collected handwritten datasets from 8 different Indian languages and developed HTR for Indic scripts. In this chapter, we introduce the IIIT-HW datasets. With these datasets, we hope that research in Indic HTR can advance further and such systems can be put to use in real life. In later chapters, we discuss recognition results on these datasets. We also conduct studies to understand the similarities between the scripts and improving recognition using minimal data, transfer learning approaches etc.

4.1 Motivation

In the last two decades, large digitization projects converted paper documents and ancient historical manuscripts into digital forms. However, they remain often inaccessible due to the unavailability of robust HTR solutions. The capability to recognize handwritten text is fundamental to any modern document analysis system. There is an immediate need to provide content-level access to the millions of manuscripts and personal journals, large court proceedings and also develop HTR applications to automate the processing of medical transcripts, handwritten assessments etc. In recent years, efforts towards developing text recognition systems have advanced due to the success of deep neural networks [70, 30] and the availability of annotated datasets. This is especially true for Latin scripts [9, 58, 8]. The IAM [50] handwritten dataset introduced over two decades ago, the historic George Washington [18] dataset and the RIMES [27] dataset are some of the popularly known datasets for handwritten text recognition. These public datasets enabled research for Latin HTR. Even today, these datasets are still being utilized to study handwritten data.

Compared to Latin HTR, Indic HTR remains understudied due to a severe deficit of annotated resources in Indian languages. Unlike many other parts of the world, a wide variety of languages and scripts are used in India. Therefore, collecting sizeable handwritten datasets for multiple Indic scripts becomes challenging and expensive. Existing annotated datasets for Indic HTR are limited in size and scope. They are approximately $5\times$ smaller than the Latin counterparts.

Recent progress in text recognition is mainly credited to the easy access to large annotated datasets. Through this work, we make an effort to bridge the gap between the state of the arts in Latin and Indian languages by introducing a handwritten dataset written in 8 Indian scripts. We introduce a dataset for Bengali, Gujarati, Gurumukhi, Kannada, Odia, Malayalam, Tamil, and Urdu scripts. This dataset along with the datasets IIIT-HW-DEV, IIIT-HW-TELUGU introduced by Dutta et al. [15, 16] for Devanagari and Telugu scripts provide handwritten datasets for text recognition in all ten prominent scripts used in India. We refer to this collective dataset as the IIIT-INDIC-HW-WORDS. Fig. 4.1 gives a glimpse into the new dataset and the writing style in eight different Indic scripts. This new dataset contains 868K word instances written in 8 prominent Indic scripts by 135 writers. Possibly this is the first attempt in even attempting offline handwriting in some of these scripts.

We hope that our dataset provides the much-needed resources to the research community to develop Indic HTR and to build valuable applications around Indian languages. The diversity of the IIIT-INDIC-HW-WORDS dataset makes it possible to utilize this dataset for other document analysis problems also. Script identification, handwriting analysis, and synthesis for Indian languages are examples of other problems that can be enabled using this dataset. This dataset could also be beneficial for study of script-independent recognition architectures for HTR.

This chapter also discusses and reports performance metrics on text recognition architectures for Indian HTR. We establish a high baseline on scripts presented in the IIIT-INDIC-HW-WORDS. The results and related discussion are in Section 4.4. We also study the benefit of using architectures pre-trained on other scripts. Building robust recognizers in various scripts requires a large amount of data for each script. With transfer learning from other scripts, the excessive requirements of large dataset and training time can be reduced. We also investigate the relation between script similarity and pre-training. We explore both these ideas in Section 4.6.

4.2 IIIT-HW Datasets

4.2.1 Data collection

The most common way employed to collect handwritten data is to provide a form with a printed text snippet and an empty area to write the provided text snippet. This approach works fine if the annotation is to be provided at page-level or paragraph level. However, generating line-level or word-level annotations from such forms is not an easy task owing to the text overflowing into new lines, inherent skew variations observed across different writers, overwritten words and mistakes etc.

Table 4.1: Publicly available handwritten datasets for Indic scripts and comparison to the dataset introduced in this work. All the datasets provide word-level transcriptions, except for the PBOK dataset.

Name	Script	#Writers	#Word Instances	#Lexicon
PBOK [2]	Bengali	199	21K	925
	Kannada	57	29K	889
	Odia	140	27K	1040
ROYDB [63]	Bengali	60	17K	525
	Devanagari	60	16K	1030
CMATERDB2.1 [6]	Bengali	300	18K	120
CENPARMI-U [64]	Urdu	51	19K	57
LAW [35]	Devanagari	10	27K	220
TAMIL-DB [73]	Tamil	50	25K	265
IIIT-HW-DEV [15]	Devanagari	12	95K	11,030
IIIT-HW-TELUGU [16]	Telugu	11	120K	12,945
IIIT-INDIC-HW-WORDS (This work)	Bengali	24	113K	11,295
	Gujarati	17	116K	10,963
	Gurumukhi	22	112K	11,093
	Kannada	11	103K	11,766
	Odia	10	101K	13,314
	Malayalam	27	116K	13,401
	Tamil	16	103K	13,292
	Urdu	8	100K	11,936
	8 scripts	**135**	**868K**	**97,060**

As a solution to the word-annotation and data collection problem, Kartik et al[14] propose an efficient form generation and annotation pipeline to collect handwritten data at word level. The forms consist of multiple word boxes and a QR code is associated with each word box. Each word box has a printed word and an empty area to write the printed word. As the forms are generated in a controlled manner, we have the positions of each word box and therefore we can extract the written word from these boxes. The label for these extracted words can be obtained by decoding the QR codes associated with the respective boxes. As a part of work done in [14], two datasets IIIT-HW-Dev and IIIT-HW-Tel were released. These datasets have 100k annotated words. In this work, we extend the IIIT-HW datasets by adding handwritten datasets in 8 other Indic scripts. We employ the same approach of form generation and annotation to collect handwritten data.

4.2.2 IIIT-INDIC-HW-WORDS

This section introduces the IIIT-INDIC-HW-WORDS dataset consisting of handwritten image instances in 8 different Indic scripts. The scripts present in the collection are Bengali, Gujarati, Gurumukhi, Kannada, Malayalam, Odia, Tamil and Urdu. Indian scripts belong to the Brahmic writing system. The scripts later evolved into two distinct linguistic groups: Indo-Aryan languages in the Northern India and

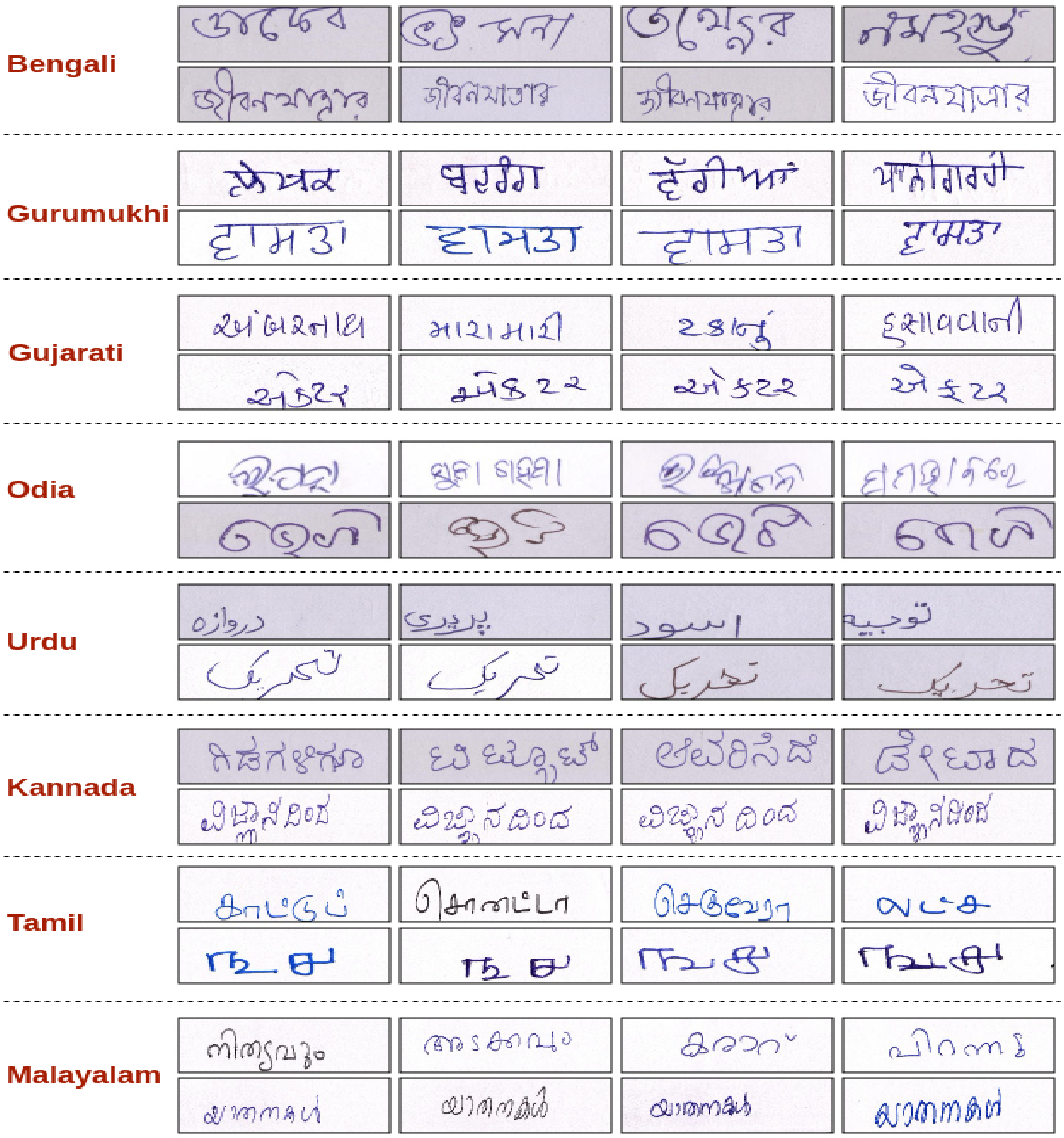


Fig. 4.1: Word instances from our IIIT-INDIC-HW-WORDS dataset. Out of the 10 major Indic scripts used, we present datasets for 8 scripts: Bengali, Gujarati, Gurumukhi, Kannada, Malayalam, Odia, Tamil, Urdu and complement the recent efforts in Dutta et al. [15] and [16] for Hindi and Telugu. For each script, row 1 shows writing style variations for a specific writer across different words. Row 2 images presented in a specific script block show four writing variations for a particular word.

Table 4.2: Statistics of IIIT-INDIC-HW-WORDS dataset and IAM dataset.

Dataset	Script	Word Length (Avg)	#Instances Train set	Val set	Test set
IAM	English	5	53,838	15,797	17,615
IIIT-INDIC-HW-WORDS	Bengali	7	82,554	12,947	17,574
	Gujarati	6	82,563	17,643	16,490
	Gurumukhi	5	81,042	13,627	17,947
	Kannada	9	73,517	13,752	15,730
	Odia	7	73,400	11,217	16,850
	Malayalam	11	85,270	11,878	19,635
	Tamil	9	75,736	11,597	16,184
	Urdu	5	71,207	13,906	15,517

Dravidian languages in the South India. Out of the 8 scripts discussed in this work, 5 scripts are of Indo-Aryan descent, and 3 are of Dravidian descent. Bengali, Gujarati, Gurumukhi, Odia, and Urdu belong to the Indo-Aryan family. Kannada, Malayalam, and Tamil are Dravidian languages. Indian scripts run from left to right, except for Urdu. Table 4.2 shows the statistics of this dataset. In the dataset, more than 100K handwritten word instances are provided for each script. The entire dataset is written by 135 natural writers aged between 18 and 70. More details about the dataset are mentioned on the dataset web-page.

Dutta et al. [16] propose an effective pipeline for the annotation of word-level datasets. We employ the same approach for collection and annotation generation for IIIT-INDIC-HW-WORDS. A large and appropriate vocabulary is selected to cover the language adequately. Participants are asked to write in a large coded space on an A4 paper. The written pages are scanned at 600 DPI using a flatbed scanner. Word images and associated annotations are automatically extracted from the scanned forms using image processing techniques. The average image height and width in the dataset is 288 and 1120. Samples that are wrongly segmented or those containing printed text, QR codes, and box borders are eliminated from the dataset. Such samples accounted for 4% of the total extracted samples. Except for removing wrongly segmented words, we do not perform any other kind of image pre-processing. Fig. 4.1 shows word instances for each of these scripts. We describe some of our observations here. The words found in Dravidian languages are longer than those present in Indo-Aryan languages. We plot the differences in word lengths between these two groups in Fig. 4.2. We also show the inter-class and intra-class variability in handwriting styles across Indic scripts in Fig. 4.1. The variable handwriting styles, noisy backgrounds, and choice of the unwanted pen makes it a diverse and challenging collection. Many samples in the dataset contain overwritten characters or poorly visible characters due to pen use. The varying background noise and paper quality also adds to the complexity of the dataset.

The text lexicon for the datasets is sampled from the Leipzig corpora collection [21]. It consists of text files with content from newspaper articles, Wikipedia links, and random websites. More than 10K unique words are sampled from these collections in each language to generate the coded pages for

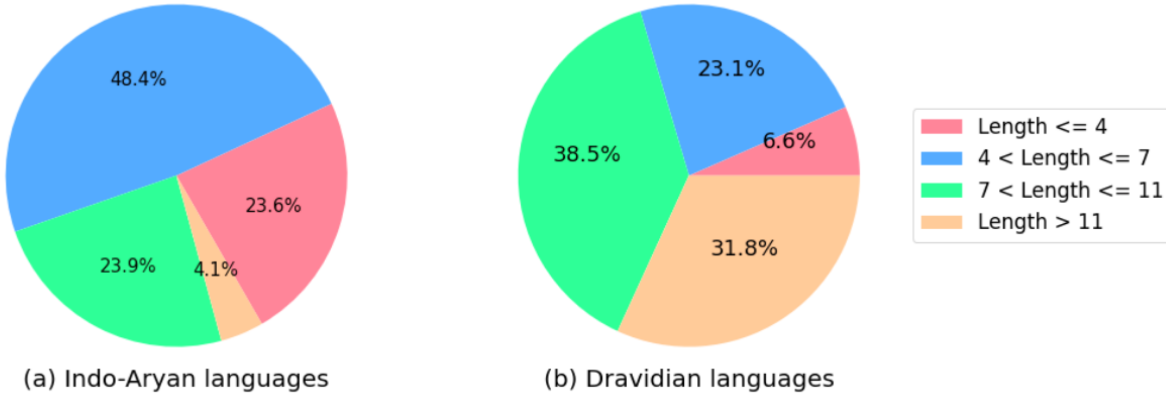
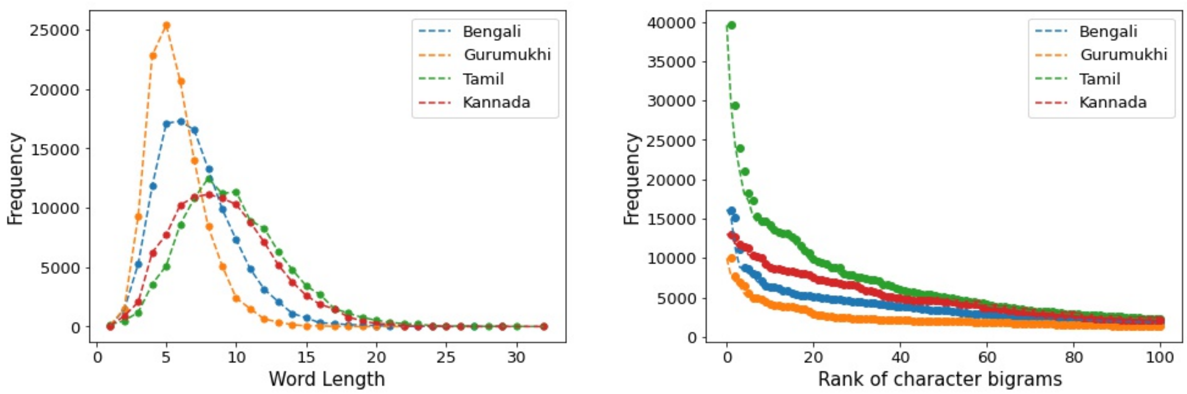


Fig. 4.2: Plots showing the variation in distribution of word lengths in IIIT-INDIC-HW-WORDS dataset for two linguistic groups: Indo-Aryan languages and Dravidian languages.



(a) Word length distribution

(b) Character bigram distribution

Fig. 4.3: Distribution plots for four scripts: Bengali, Gurumukhi, Tamil, and Kannada. We observe that above distributions for Gujarati, Odia are similar to that of Bengali script. Distributions for Urdu and Malayalam script are comparable to those of Gurumukhi and Tamil script respectively.

participants to write. The number of unique characters per script includes the basic characters in the respective script's Unicode block and some special characters. The statistics of lexicon size for the IIIT-INDIC-HW-WORDS collection are listed in Table 4.1. Fig. 4.3a shows the distribution of word length across different datasets. The plot 4.3b shows that the character bigrams follow Zipf's law [57]. We observe that the law holds true for character n-grams as well with $n = 3, 4, 5, 6$.

The annotated dataset consists of image files along with a text file containing the corresponding label information. The text labels are encoded using Unicode. We also release the train, test, and validation splits for the word recognition task. Around 70% of the instances are added to the training set, 15% to the test set, and the remaining instances comprise the validation set. We ensure that 9-12% of the test labels are out-of-vocabulary(OOV) words. Test sets with a high rate of OOV samples are challenging. The evaluated metrics on such sets inform whether a proposed solution is biased towards the vocabulary

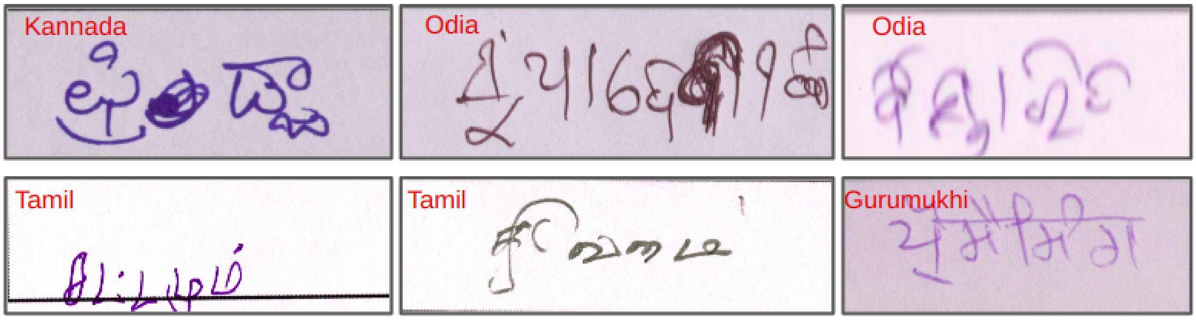


Fig. 4.4: Difficult and challenging samples from IIIT-INDIC-HW-WORDS. These instances have poor writing styles, poor visibility, and segmentation errors.

of training and validation sets. The total number of samples having out-of-vocabulary text labels in the test sets varies from 35% to 40% per script.

4.3 Challenges in Indic Text Recognition

Indian scripts are complex and challenging. Most of the Indian scripts follow the *abugida* writing system. It is a segmental writing system where the consonants and vowels are written together as a single unit. Each consonant is associated with inherent vowel. The associated vowel can be changed by adding a diacritic above, below, before, or after the consonant. This combination is referred to as *akshara* and is analogous to a character in Latin scripts, a basic unit to form words. When combining vowels with consonants, the structure of the written character changes drastically depending on the choice of vowel and consonant. In a Indian language, every phoneme is associated with a consonant-vowel and therefore, the alphabet is larger than its Latin counterparts. This is because of the various possible combinations of *aksharas*. Similarly, other concepts like lengthening of consonants, conjunctive forms for consonants and their combinations with vowels are used to express complex words in Indic scripts. For better understanding, the readers can refer to discussion and examples mentioned in [32]. Indic HTR has to overcome the difficulty of recognizing the words that follow these complex concepts along with overcoming common challenges like varying writing styles, distortion, and spelling errors.

4.4 Baseline for Text Recognition

In the previous chapter, we describe a generic handwritten text recognition pipeline and its components. In this chapter, we use the discussed HTR pipeline to establish a baseline on the IIIT-INDIC-HW-WORDS dataset. In this section, we discuss and study the different stages of the pipeline for Indic HTR. We duplicate the pipeline diagram in Fig. 4.5 for easier referencing.

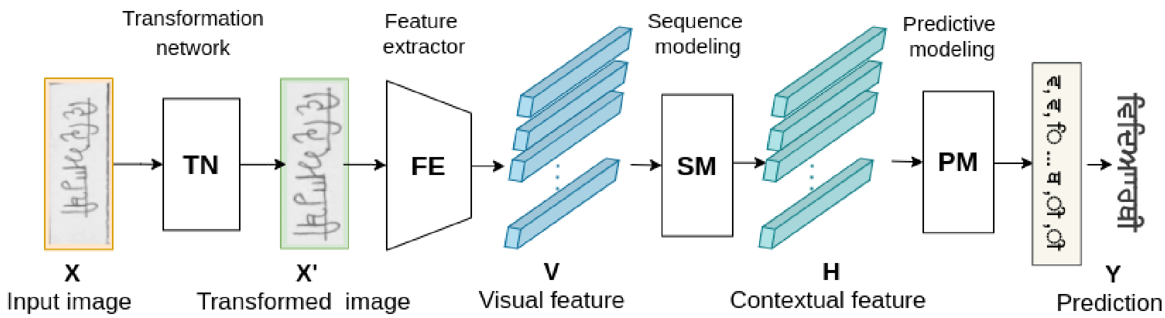


Fig. 4.5: Generic pipeline for text recognition. Here, we demonstrate the flow of a sample image **X** through the pipeline to generate a text prediction **Y**. The sample shown here is written in Gurumukhi script.

- **Transformation Network(TN):** Diverse styles observed in handwriting data are a significant challenge for HTR. In this work, we experiment with two types of rectification, affine transformation (ATN), and thin-plate spline transformation(TPS). Affine STN applies a transformation to rectify the scale, translation, and shear. TPS STN applies non-rigid transformation by identifying a set of fiducial points along the upper and the bottom edges of the word region.

- **Feature Extractor (FE):** For Indic HTR, we study two commonly used CNN architecture styles: VGG [70] and RESNET [30]. VGG-style architecture comprises multiple convolutional layers followed by a few fully connected layers. RESNET-style architecture uses residual connections and eases the training of deep CNNs. Shi et al. [69] tweak the original VGG architecture so that the generated feature maps have larger width to accommodate for recognition of longer words. Dutta et al. [15] introduce a deeper architecture with residual connections. We study both these architectures in this work. We refer to these as HW–VGG and HW–RESNET.

- **Sequence Modeling(SM):** Bidirectional LSTM(BLSTM) is the preferred block to compute the new feature sequence H as it enables context modeling from both directions. Due to its success [69, 58, 17, 4], we use a 2 layer BLSTM architecture with 256 hidden neurons in each layer as the SM module in our experiments.

- **Predictive Modeling(PM):** We use CTC [24] layer for generating predictions. The CTC objective function was discussed in detail in previous chapter. Treating Indic HTR task as learning to recognize the sequence of *aksharas* can be restrictive as the total number of unique *aksharas* in a script can exceed over 20,000 characters[46]. To overcome this hurdle, we recognize Indic text as a sequence of Unicode symbols, which reduces the unique symbol count by 20 times roughly.

- **Data Augmentation:** For Indic htr, the augmentation pipeline consists of affine, elastic, perspective transformations along with brightness and contrast transformation. Brightness and contrast augmentation is applied to learn texture and background invariant features. The affine transformation parameters are restricted by ± 5 degrees for rotation and and shearing. The elasticity coefficient for Indo-Aryan and Dravidian languages is set to 0.3 and 0.2 respectively.

4.5 Experiments and Discussion

In this section, we evaluate and discuss four architecture combinations in the HTR pipeline. Through this setup, we aim to determine the best model for the Indic HTR task. The model architectures are listed in Table 4.3. For every upgrade done from M1 to M4, we introduce better alternatives in a single stage of the HTR pipeline. We observe the improvement introduced by a specific module. The best performing architecture is identified from the combinations while considering the trade-off between evaluation metrics and computational complexity. We evaluate the performance of these architectures on two datasets: Tamil and Bengali. The Tamil dataset belongs to the Dravidian linguistic group, and the Bengali dataset belongs to the Indo-Aryan group.

Table 4.3: Results on Bengali and Tamil script in IIIT-INDIC-HW-WORDS dataset for common HTR architectures. TN-FE-SM-PM refers to the specific modules used for text recognition. BLSTM-CTC is used for SM-PM stages in the experiments.

#	TN	FE	CER		Params
			Bengali	Tamil	x10^6
M1	None	HW–VGG	11.48	7.25	8.35
M2	ATN	HW–VGG	9.41	5.99	8.38
M3	ATN	HW–RESNET	5.47	1.47	16.54
M4	TPS	HW–RESNET	5.22	1.38	17.15

We compare the alternatives in TN and FE stages of the HTR pipeline against character error rate(CER) and total number of parameters associated with the architecture. The results obtained are presented in Table 4.3. For feature extraction stage, replacing HW–VGG with HW–RESNET architecture(M2 $\rightarrow$ M3) increases the number of parameters by $2\times$. However, architecture M3 significantly improves the error rate by at least 4%. As the number of layers are increased, architecture M3 benefits from the network's increased complexity and representation power.

Changing the transformation network from ATN to TPS(M3 $\rightarrow$ M4) reduces the error rate by minor margin only. We conduct another experiment to understand the effectiveness of TN. For this study, we train two architectures with and without TN for different sizes of training data. We train the architectures with 10K, 30K, 50K, and 82K samples for Bengali script. The remaining stages of the pipeline are chosen as HW–RESNET-BLSTM-CTC. The comparison plot in Fig. 4.6 shows that the WER for the TPS variant reduces significantly when the training datasets have limited samples. With limited training data, the TN stage is crucial to build robust HTR. The introduced changes in TN and FE stages increases the architecture's complexity resulting in smaller error rates.

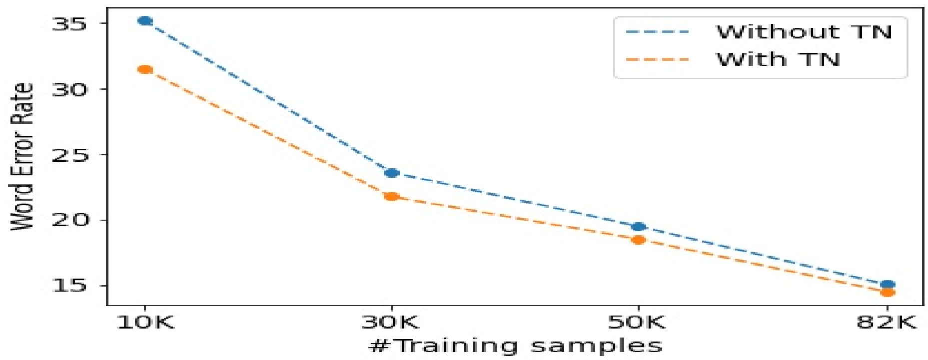


Fig. 4.6: Word error rate(WER) vs. training size for two architectures: with TPS TN and without TN. The FE, SM, PM stages are fixed as HW–RESNET, BLSTM, CTC. The results are shown for Bengali script in IIIT-INDIC-HW-WORDS dataset.

With the HTR pipeline as TPS-HW–RESNET-BLSTM-CTC, we train and evaluate its performance on all the scripts in the IIIT-INDIC-HW-WORDS dataset. The samples are augmented randomly with affine, elastic, and color transformations. The model is fine-tuned on the pre-trained weights from the IAM dataset. To this end, we train the architecture on IAM dataset as well and the WER and CER obtained are 13.17 and 5.03 respectively. The obtained performance metrics are reported in Table 4.4. We report error rates on two sets; the test set and its subset containing only OOV words. From the table, the error rates observed for Kannada script is the lowest and the error rates for Urdu script are the highest. It is interesting to observe similar performances within the Indo-Aryan group and Dravidian group. We also note that the Urdu dataset is more challenging than other datasets due to high error rates, despite having fewer writing styles. Fig. 4.7 shows the predicted text for selective samples from the IIIT-INDIC-HW-WORDS dataset. The model fails to recognize the samples presented in the third column by less than two characters. The recognized text for these samples is incorrect due to confusing character pairs or predicting an extra character at the end of the text string. For the samples presented in the fourth column, the longer length of words causes the recognizer to make mistakes and generate shorter predictions.

4.6 Analysis

Training deep neural architectures for a specific task requires optimization of millions of parameters. Due to their large size, training from randomly initialized weights requires a lot of time and hyper-parameter tuning. Pre-trained weights computed for other tasks are generally used to initialize new networks [13, 81]. This is also referred to as transfer learning. This technique is especially meaningful in HTR where the number of training samples is limited in a specific domain or script [34, 23]. Given a

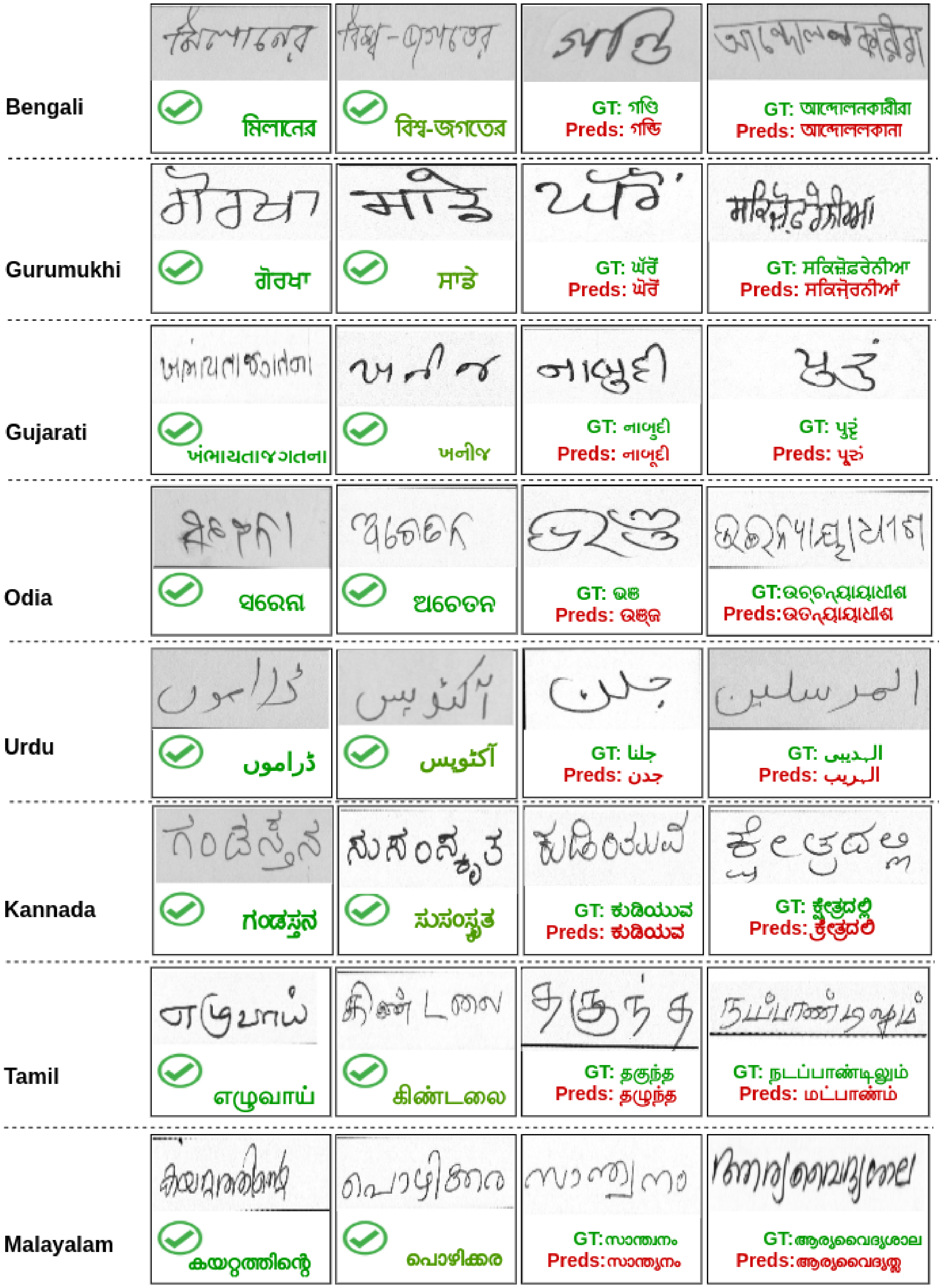


Fig. 4.7: Qualitative results on the IIIT-INDIC-HW-WORDS dataset. Word images are converted to grayscale and forwarded to the recognizer. Column 1 & 2 present correctly predicted samples. Column 3 & 4 shows incorrect predictions. GT and Preds refer to ground truth and predicted label respectively.

Table 4.4: Results on IIIT-INDIC-HW-WORDS dataset. Metrics listed are computed on the test set and out-of-vocabulary(OOV) test set. CER is character error rate and WER is word error rate. Characters refers to number of unique Unicode symbols in each dataset.

Script	Characters	Test		OOV Test	
		CER	WER	CER	WER
Bengali	91	4.85	14.77	3.71	16.65
Gujarati	79	2.39	11.39	5.27	25.8
Gurumukhi	84	3.42	12.78	6.85	23.97
Odia	81	3.00	14.97	4.97	23.40
Kannada	83	1.03	5.90	1.59	9.39
Malayalam	95	1.92	9.85	2.06	10.6
Tamil	72	1.28	7.38	1.88	8.25
Urdu	81	3.67	15.00	9.33	33.23

dataset with limited training samples, performing transfer learning from a large dataset eases the training and improves the performance.

For HTR domain, Bluche et al. [8] discuss the advantages of training recognizers using big multi-lingual datasets in Latin scripts. Dutta et al. [16] show that using pre-trained architectures for fine tuning on Indic datasets improves the performance. They utilize weight parameters learnt on the IAM dataset to reduce the error rate on the IIIT-HW-DEV and the IIIT-HW-TELUGU datasets. Whereas, recent research [29] points out a network trained from scratch for long intervals is not worse than a network finetuned on pre-trained weights for detection and segmentation task. In this study, we explore whether pre-training is useful within the context of handwritten data across multiple scripts. We use three scripts to conduct these experiments: English, Devanagari and Telugu. We also explore the utility of pretrained architectures with varying number of training samples.

In this study, we explore the training architectures using randomly initialized weights, and pre-trained weights from IAM, IIIT-HW-DEV and IIIT-HW-TELUGU datasets. We also explore these pre-training strategies for varying sizes of training data. We present the results on the Bengali data in the IIIT-INDIC-HW-WORDS dataset. The networks are trained for same number of iterations using Adadelta optimization method. The final M4 architecture is trained with data augmentation in this experiment and the results obtained are presented in Table 4.5. The benefit of using pre-trained architectures for smaller training datasets is notable and gives an improvement of 2.75% in word error rate(WER). As the available training data increases, the reduction in error rates is not apparent. We observe that a randomly initialized network is not necessarily worse at recognition than the network using pre-trained weights for initialization.

Through the above experiment, we also conclude that language similarity is not a major contributing factor to performance improvement for HTR task. The datasets used in the study are written in Latin, Devanagari, and Telugu script. The corresponding datasets for these scripts are IAM, IIIT-HW-DEV, IIIT-HW-TELUGU. Bengali and Devanagari scripts share similarities and both the scripts belong to the Indo-

Table 4.5: Effect of pre-training with varying training size. The results shown here are computed on the IIIT-INDIC-HW-WORDS Bengali dataset.

Training size	Pre-trained Weights	WER	CER
10K	None	31.54	9.64
	IAM	29.28	8.98
	IIIT-HW-DEV	28.73	8.94
	IIIT-HW-TELUGU	**28.79**	**8.71**
30K	None	21.76	6.88
	IAM	20.37	6.43
	IIIT-HW-DEV	**20.10**	**6.36**
	IIIT-HW-TELUGU	20.35	6.38
50K	None	18.08	5.67
	IAM	**17.23**	**5.49**
	IIIT-HW-DEV	18.30	5.72
	IIIT-HW-TELUGU	17.8	5.67
82K	None	15.34	4.97
	IAM	**14.77**	**4.85**
	IIIT-HW-DEV	15.35	4.97
	IIIT-HW-TELUGU	15.0	5.04

Aryan group. However, results show that pre-training from similar language outperforms other pre-training techniques only for one of the cases and also by very small margin. Interestingly, the pretrained weights from the IAM dataset provide best results as the training data increases to 82K samples.

We confirm the above claim with another experiment. Here, we train recognizers for Odia, Urdu and Malayalam in the IIIT-INDIC-HW-WORDS dataset with pre-trained weights from the Latin script IAM dataset and the Bengali script IIIT-INDIC-HW-WORDS dataset. The results obtained are compared in the Fig. 4.8. From the plot, we observe that IAM pre-training produces better results than the Bengali pre-training approach for both Odia and Urdu. On the contrary, among the three target scripts Bengali and Odia have similar writing styles.

From this brief study, we conclude that pre-training is especially meaningful for handwritten tasks when the available training data is limited. Therefore, pre-trained architectures can be utilized to build recognizer for historical data or regional scripts. Transcriptions for a few samples are sufficient to fine-tune the recognizers for such target tasks. We also observe that pretraining from similar scripts does not supplement the training process for Indic scripts. Both experiments conclude that the IAM pretrained model is more effective for pretraining and the underlying reason for this effect could be due to greater number of writing styles seen in the IAM dataset.

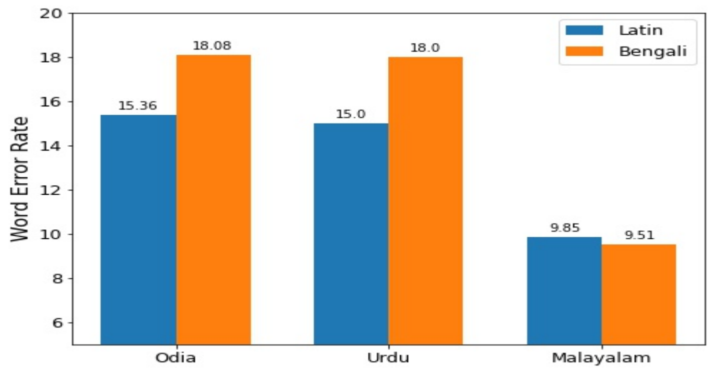


Fig. 4.8: Comparison between Latin script vs. Bengali script pre-training for Odia, Urdu and Malayalam recognizers. The Latin script is trained on the IAM dataset and the Bengali script recognizer is trained on the IIIT-INDIC-HW-WORDS Bengali dataset.

4.7 Summary

In this chapter, we introduce a collective handwritten dataset for 8 new Indian scripts: Bengali, Gujarati, Gurumukhi, Kannada, Malayalam, Odia, Tamil, and Urdu. We extend our earlier efforts in creating Devanagari [15] and Telugu [16] datasets. We hope the scale and the diversity of IIIT-INDIC-HW-WORDS dataset in all 10 prominent Indic scripts will encourage research on enhancing and building robust HTRs for Indian languages. We believe it is essential to continue improving Indic datasets to enable Indic HTR development, and therefore, future work will include enriching the dataset with more writers and natural variations. Another important direction is to create handwritten data at line, paragraph and page level.

Chapter 5

Keyword Spotting and retrieval

5.1 Introduction

In this chapter, we introduce an Embedding-Based Retrieval(EBR) framework that enables search over handwritten collections. EBR approaches use deep embeddings to solve a given retrieval task. Deep embeddings are points in a high-dimensional space(embedding spaces) corresponding to raw data points. This embedding space extracts and captures similarities observed in raw data points. It does so by generating meaningful clusters in the embedding space. The best part is that definition of similarity can be customized to capture semantic similarity, content similarity, style similarity, etc. Custom objective functions can be defined for this purpose. These functions ensure the generation of deep embeddings that can bring out meaningful relations and make them machine understandable. The embeddings are then used for similarity search operations to perform retrieval. EBR approaches are becoming increasingly popular for large-scale search in gigantic data-driven applications like Google Search and Facebook Search. As learning a holistic representation from raw data is vital in an EBR task, machine learning approaches like representation learning can be employed.

In the context of handwritten text spotting, representation learning is utilized to match handwritten images to a given query based on the textual content. Work done in [3, 72, 45] discusses unique ways of embedding learning for handwritten text. These spotting approaches present promising results on the standard handwritten datasets. Earlier chapter 2 discusses the related background for text spotting. Therefore, we are motivated to use an EBR approach for retrieval. EBR approach also allows for tweaking and addition of language embeddings like BERT embeddings to further improve the nature of search.

The following section defines the scope of the problem statement, its challenges, and our proposed setup for end-to-end retrieval followed by system implementation details. Our framework is implemented to demonstrate retrieval in novel Indian collections. The details and performance analysis are presented in the later sections. We also conduct experiments to compare our EBR framework to baseline and other EBR frameworks. We establish that our simple framework is robust and better than baseline frameworks.

5.1.1 Scope and Challenges

Text spotting is a widely studied topic for standard datasets that contain the necessary data labels needed for data-driven models. Evaluation metrics tracked for text spotting in various works[45, 72, 80] report mean average precision (mAP) scores greater than 95%. However, these reported metrics are calculated on a subset (test split) of the annotated dataset. In other words, retrieval performance is generally evaluated in a single domain. Additionally, such annotated datasets are relatively neat and legible. On the contrary, natural handwritten collections are more complex than these datasets and there exists a huge domain gap between these collections and the annotated datasets. This begs the question on how the current approaches apply to real-world settings, where the required labels are primarily unavailable. Due to the domain gap, the spotting and retrieval performance drops in an unseen document set. Hence there arises a need to study and evaluate existing approaches for retrieval tasks in unexplored document collections.

Earlier in the thesis, we established that the intrinsic and extrinsic variations seen in handwriting are tricky to generalize. And this causes a domain gap between standard datasets and real-world document collections. Therefore, models developed for benchmark datasets show a performance drop on new and unseen documents. Finetuning existing spotting models for new data is not always feasible as handwriting annotation is expensive and requires manual labour with expertise in multiple scripts. This chapter discusses the problem of text search and spotting in generic handwritten documents that are unlabeled. We discuss the key components required to build a large-scale handwritten search application. We use robust and discriminative feature extractors, efficient indexing methods, and fast feature matching methods to build a retrieval pipeline. Robust feature extractors generate embeddings invariant to writing styles, paper textures and degradation.

Another limitation of existing works is the lack of studies and discussion of non-Latin scripts. Massive manuscript collections exist in many non-Latin scripts and some of them are more challenging than the Latin script. For example, over a million manuscripts written in various Indic scripts are dispersed worldwide. Other issues like poor image quality and degradation, illegible handwriting, complex layouts, and open vocabulary increase the difficulty level for spotting keywords from unexplored collections. We study and evaluate the performance of our framework on untranscribed collections in English and 2 Indic scripts: Malayalam and Bengali. This setup can easily be extended to other Indic scripts. Another crucial aspect of any retrieval framework is the search time and memory requirements ensuring its scalability. Therefore, we study and discuss efficient approaches to improve retrieval speed.

We explore two input modes for a search operation in this work: textual search and exemplar search. Textual search is the most commonly used mode of input in search engines. Additionally, exemplar search can be useful in the handwriting domain as it simplifies the search process and has additional applications like searching for unknown or unidentified symbols. These two search modes are referred to as query-by-string(QbS) and query-by-example(QbE) settings in the context of keyword spotting.

As our work is focused on retrieval from unlabeled collections, handwritten document collections from public digital libraries are used to demonstrate the efficiency of our retrieval pipeline. These col-

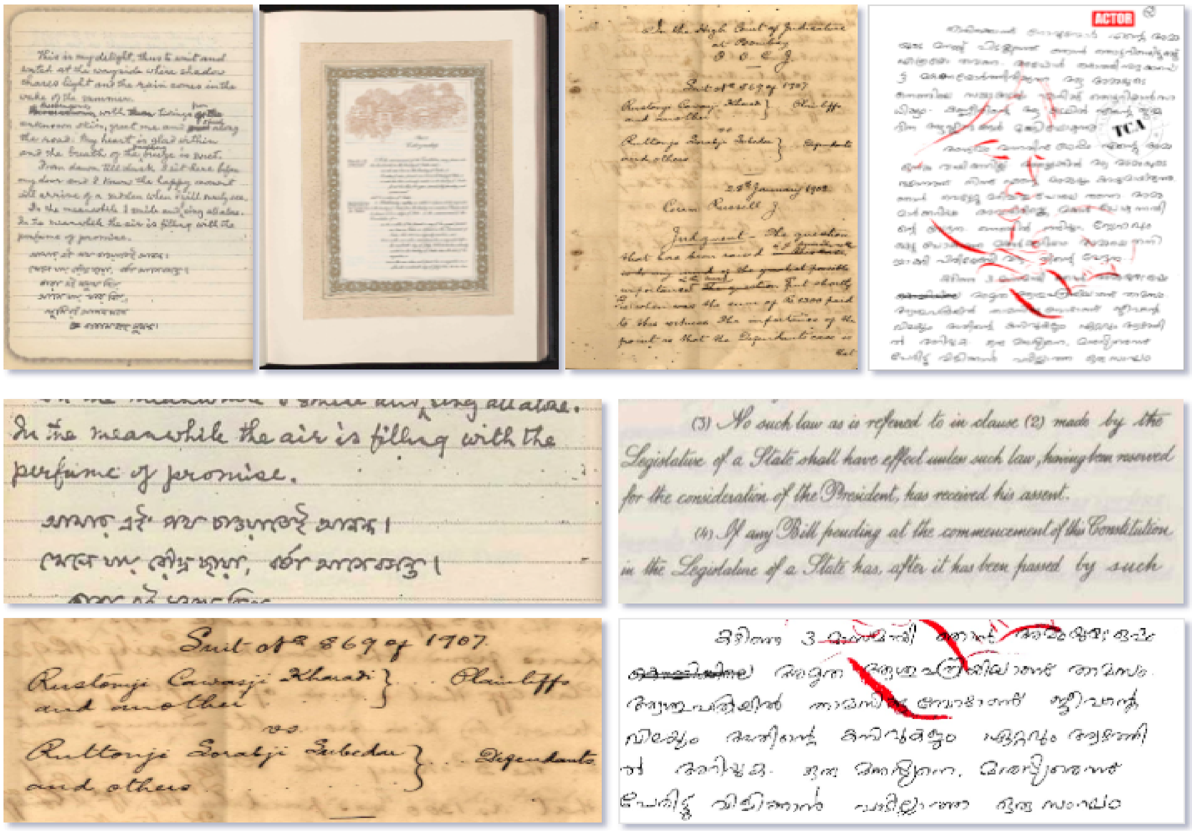


Fig. 5.1: Sample snippets from the historic and contemporary Indian handwritten collections: *Tagore's papers*, *Constitution of India manuscript*, *Bombay High Court judgements*, *Mohanlal writings*. Malayalam writing style is shown in bottom right image and Bengali writing style is shown in top left image. Challenges in these collections involve poor image quality, ink bleed, complex backgrounds and layouts.

lections contain a total of 2,957 pages and 313K words. Fig.5.1 shows snippets from chosen collections discussed in this work. Search on these specific records is difficult as they come with major issues like poor image resolution, unstructured layouts, different scripts in a single page, ink-bleed, ruled and watermark backgrounds. We address these challenges and discuss a retrieval approach developed for these unseen collections. We utilize complementary modules and spotting approaches to tackle some of these challenges. We perform experiments on these collections to study the efficiency of our retrieval pipeline for searching across large document collections. Finally, we present a real-time handwritten document retrieval demonstration[1] for handwritten collections from India.

The major contributions discussed in this chapter are as follows:

i. An embedding based framework for searching and retrieval from handwritten collections.

ii. Study and discussion on the approaches towards generic keyword spotting that can handle unseen words and unseen writing styles.

[1]Demo video: http://cvit.iiit.ac.in/images/Projects/hwsearch/hwsearch_demo.mp4

iii. Evaluation of efficient indexing, and query matching methods.

iv. A demonstration to validate our end-to-end retrieval pipeline on 4 collections written in English, Bengali and Malayalam.

5.2 Proposed setup

This section discusses two major aspects of a spotting framework: feature extraction and feature matching. We use deep embeddings as features. These embeddings are fixed-length holistic representations for handwritten word images and text strings. Section 5.2.1 discusses this in more detail. Fixed-length representations allow for the application of faster retrieval methods like similarity search. We discuss our non-exhaustive similarity search approach for feature matching and ranked retrieval in section 5.2.2.

5.2.1 Embedding learning

Spotting a given keyword in an unseen document can be achieved by learning holistic feature representations for handwritten text. These representations must be invariant to writing styles, document degradation and poor image resolution. At the same time, the feature learning method should also generate discriminative representations for an open set of vocabulary. To achieve these goals, we employ the end-to-end deep embedding network, HWNetV2, discussed in [43]. This network learns a common subspace of embeddings for both word images and text strings. In other words, a text string and word image containing the same string will have similar representations in the embedding space. We utilize these embeddings as features for indexing and matching.

The overview of this architecture is shown in Fig. 5.2. The network consists of two streams: the real stream and the label stream. Real stream processes the input handwritten word images to compute a feature representation using a deep CNN proposed in [45]. Label stream has two modules: PHOC [3] and a CNN feature extractor. The input to these blocks is a text string and the corresponding synthetic image, respectively. These features are concatenated to form a unified label stream embedding. The real stream and label stream embeddings are forwarded to an embedding layer consisting of a multilayer perceptron (MLP). This layer learns to map these feature representations from two different modalities to a common embedding/representation space. This deep network is optimized using two objective functions: (i) maximize the similarity between the real stream and label stream embeddings, (ii) predict the class label of the real stream image from the train vocabulary.

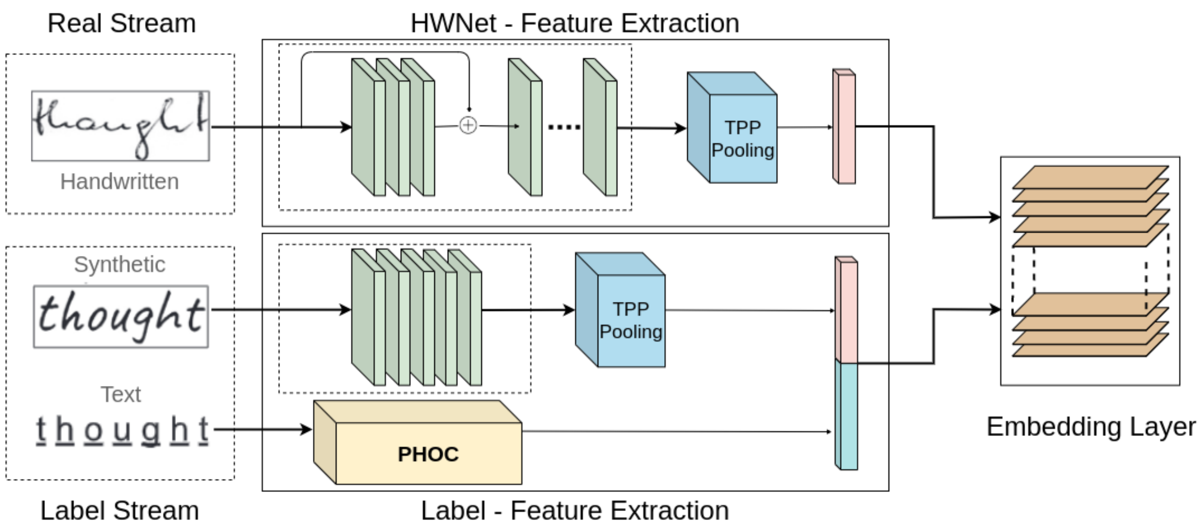


Fig. 5.2: Overview of the HWNetv2 embedding network for generating holistic handwritten text features. Real stream handles the word image input and label stream handles the text string input. Embedding layer in the network is responsible for mapping the image features and text string features to a common subspace.

5.2.2 Indexing & Retrieval

The setup employed for searching in a document collection is discussed here. Fig. 5.3 shows an overview of the retrieval pipeline. Given a historic collection, the document images are forwarded to the CRAFT [5] text detector. The output from the detector is processed to extract valid text regions from the images, which are further forwarded to the embedding network. The computed embeddings are indexed to form an embedding index for a collection. This index is used for matching an input query with the textual content of the collection. The bounding boxes obtained for text regions from text detector are also indexed to form a position and page index. This index is used for retrieving appropriate document images based on the index matching results.

Input query to the proposed setup can either be a word image or a text string. Querying using a text string is referred to as QbS(Query by String) and QbE(Query by Example) refers to querying using a word image. Similarity search is performed using the euclidean distance metric to retrieve relevant documents. The smaller the distance, the higher the similarity and rank in the result list. Using brute force search in massive indexes is costly as it requires intensive compute power. We use approximate nearest neighbor approaches as these methods are non-exhaustive. This reduces the retrieval time for a given query and makes our framework efficient. Additionally, fast and efficient implementation of the search algorithm is used to improve the latency of the framework. The next section discusses more implementation details about the entire retrieval framework.

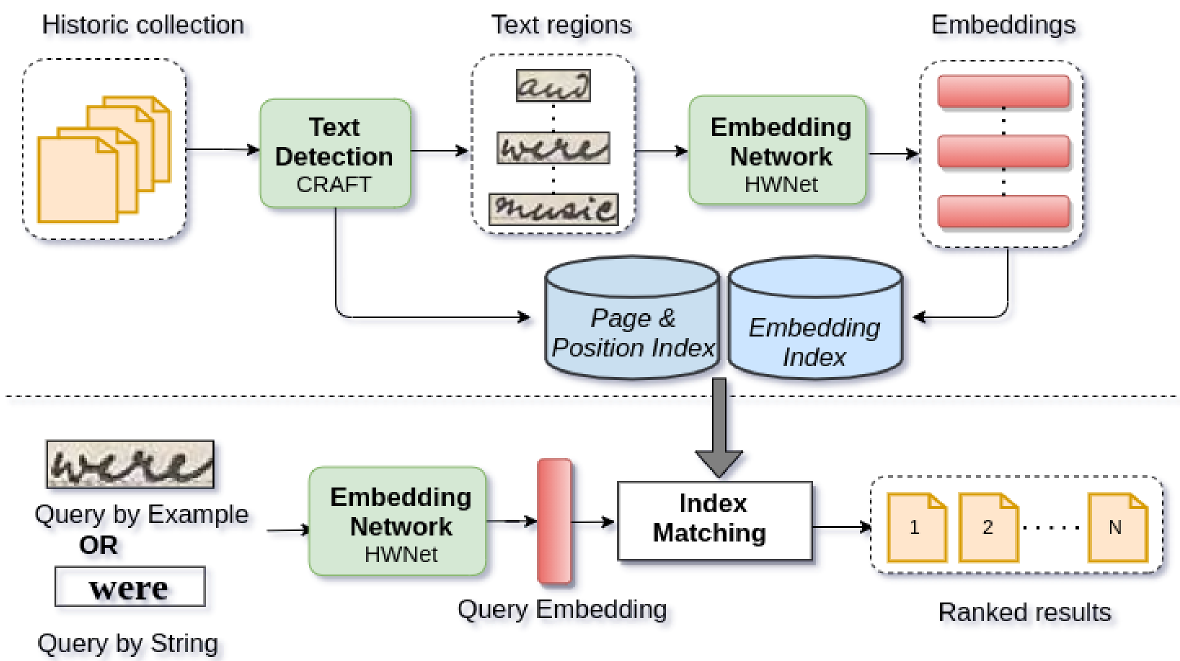


Fig. 5.3: Overview of the spotting and retrieval framework. (Top) Processing and indexing a handwritten collection to enable search operations. (Bottom) The flow of a query across the retrieval framework to retrieve relevant results.

5.3 Implementation Details

A typical word spotting framework consists of a feature extraction module, an offline module for indexing a document collection and a querying module to retrieve relevant results from the chosen collection. We describe each of these modules in the sub-sections below, along with the characteristics of the retrieval pipeline.

5.3.1 Characteristics

Our retrieval pipeline is designed so that the user can search within a specific collection with a single query. We believe that this is a reasonable assumption as the domains of the collection vary significantly and a user looking for specific information knows which collection to choose for querying. By making this choice, we reduce the search index size and, therefore, improve the retrieval time. Another characteristic of the pipeline is that querying can be done through both text strings(QbS) and word image examples(QbE). The designed document retrieval pipeline is generic and can be applied to any collection. Our proposed setup can deal with unseen document images with reasonable confidence. Efficient representation learning enables this zero-shot retrieval from unexplored collections. In our setup, labeled handwritten data is only used during training the spotting module. The HWNetv2 modeled in the training phase is utilized for performing search operations.

5.3.2 Embedding Network

The HWNetv2 network is trained using word-level handwritten datasets available publicly. IAM [50], GW [61] datasets are used for training English embeddings. For Bengali and Malayalam scripts, we use IIIT-INDIC-HW-WORDS [22] dataset. The network has millions of parameters and proper training requires training data bigger in size than these public datasets. To overcome this issue, we make use of data augmentation methods. Affine, elastic and color transformations are used to augment the training datasets. These transforms are used majorly for handwriting tasks as they can imitate the variations seen in handwriting. We also use the IIIT-HWS [44] synthetic dataset containing 1 million word images rendered using handwritten style fonts for pretraining the embedding network. The network is trained using stochastic gradient descent with an initial learning rate of 10^{-2} and a batch size 8. A scheduler is used to reduce the learning rate further as the training progresses.

5.3.3 Feature extraction

A collection is processed using Otsu thresholding to binarize and reduce the impact of different backgrounds and textures on the retrieval pipeline. We utilize a pretrained deep network CRAFT [5] to extract words from handwritten collections for text detection. This network is trained in a weakly supervised manner with synthetic data and pseudo ground truths on scene text datasets. Although CRAFT is trained on the scene text images, we observe a reasonable detection rate for handwritten documents in Latin and Indic scripts. We believe that the underlying idea behind the CRAFT detector makes the network robust to detect handwritten text. The detector works by localizing individual character regions and linking closer character regions to form words. As the detection network is used in a zero-shot setting, the text bounding boxes have a certain degree of error due to over-segmentation or under-segmentation. The HWNetv2 embeddings help mitigate these issues implicitly. Extreme affine augmentation is applied on labeled images during training to create similar learning examples for the HWNetv2 network. This transformation helps to overcome segmentation errors during test time. The extracted text image regions are forwarded to the pretrained HWNetv2 to extract features.

5.3.4 Retrieval

The computed embeddings for a collection are indexed using an inverted file index(IVF). We use the network embeddings without any quantization to create inverted indexes. While indexing a collection, the embeddings are clustered to identify Voronoi cells and their centroids, which are the representatives of the cells. For an input query, the computed embedding is used to search for top N similar Voronoi cells by matching the query embedding to the centroids. The identified Voronoi cells are matched to all the embeddings in these top N cells to obtain the top k closest embeddings. Documents corresponding to these embeddings are the top k relevant matches for the query. Embeddings corresponding to invalid text regions are eliminated. This further reduces the index size and is useful for scaling the pipeline.

Invalid text regions are over-segmented or under-segmented text regions. Additionally, small text regions are also pruned as they mostly correspond to stop words. They are also unhelpful in keyword based retrieval. Document images in a collection are segregated into different script folders to extract HWNetv2 embedding from the corresponding embedding network.

We use FAISS [36] to make efficient indexes and perform search operations. In our experiments, we cluster the embeddings into varying cell sizes depending on the total embeddings for a collection. 100 cells are used with size $< 210K$, 50 cells for $< 100K$ size and 25 cells for $< 50K$ embeddings. We choose these values based on empirical observations and the size of the datasets. We report the computation time and memory requirements in the results section.

5.3.5 Query processing

Input query is forwarded to the pretrained embedding network. Image-based queries are resized and normalized before forwarding them to the real stream of the embedding network. Text-based queries are processed to render a synthetic image containing the query string. This rendered image and the text string are forwarded to the HWNetv2 label stream. The computed query embedding is indexed using the above algorithm to compute relevant index keys and the corresponding distance between the query and matching embeddings. The index keys obtained from similarity matching are used to retrieve document ids, keyword position details from the page and position index. Relevant lines associated with the spotted query word are cropped from the corresponding documents using these positions. For this, the bounding box of the spotted query is extended along the image width and this selected area is presented as relevant lines for the input query.

5.4 Experiments & Results

5.4.1 Training Phase

The retrieval framework is developed for documents written in English, Bengali and Malayalam. The HWNetv2 embeddings are learnt for these languages. The mean Average Precision(mAP) obtained on three training datasets are reported in Table 5.1. Note that the high evaluation scores are reported on rather clean, carefully collected datasets. This is unlike a real, practical setting where the unknown handwriting, new vocabulary and unseen layouts are prevalent.

5.4.2 Evaluation datasets

Four collections from public websites and digital libraries are chosen to evaluate our pipeline: a collection consisting of handwritten poems and plays written by Rabindranath Tagore, a historical collection of Bombay High Court law judgements from the year 1864, a modern collection of handwritten blogs by an actor and a handwritten manuscript of Constitution of India(CoI). Source links for these

Table 5.1: Evaluation metrics on the test split of FAISS, IIIT-INDIC-HW-WORDS dataset. Evaluation done for both QbS and QbE settings. Full and OOV test refers to the complete test split and out-of-vocabulary test split respectively.

Dataset	QbS mAP		QbE mAP	
	Full Test	OOV Test	Full Test	OOV Test
IAM (English)	0.93	0.92	0.93	0.93
IIIT-INDIC-HW-WORDS (Malayalam)	0.99	0.98	0.99	0.99
IIIT-INDIC-HW-WORDS (Bengali)	0.95	0.95	0.95	0.94

documents are linked in Table 5.2. The table also lists the scripts, total pages and estimated words in these collections. Fig. 5.1 shows sample blocks from these collections. The pages in these collections have printed text, handwritten text, watermarks and backgrounds, illegible text due to degradation, and difficult writing styles. Our retrieval framework is capable of retrieving meaningful results for most of the queries despite these challenges. The low image resolution in CoI manuscript and Tagore's papers is also handled implicitly without nedding additional processing.

Table 5.2: Digital handwritten collections demonstrated and evaluated in this work. Digital sources are linked in the collection title column.

Collection title	Script	Pages	Words (est.)
Bombay High Court(BHC) records	English	1,964	207,028
Constitution of India(CoI) manuscript	English	479	42,040
Tagore's Papers(TP)	English, Bengali	125	10,990
Mohanlal Writings(LAL)	Malayalam	389	53,111

5.4.3 Evaluation and Discussion

This section discusses the results obtained on the four collections mentioned above. Unlike the training datasets mentioned in Table 5.1, these collections are unexplored and unlabeled. Therefore, reporting exhaustive evaluation metrics is not feasible. Labeled data is a prerequisite to compute mAP. In place of mAP, we report the precision(P) of top-k ranked results at $k = 1, 10, 25$. This evaluation method is also followed in a similar case study discussed in [80]. This web-scale metric does not require the knowledge of all relevant instances for a given query. We pick random queries from each collection and report the top-k precision(P_k) in Tables 5.3 to 5.6. The results are reported for QbS retrieval setting in these tables. We observe similar results for QbE query retrieval as well. The retrieval framework achieves similar performance for both seen and unseen out-of-vocabulary(OOV) queries. For queries like `vacancy` and `sudden` low values of P_k at $k = 10, 25$ are reported as these are rare words seen in the collection. Evaluating on CoI manuscript for extremely rare OOV words like `Madras`, `Travancore`, and `surcharge`, the P_k for $k = 1$ is 1.00. The framework can also retrieve related terms for a given query. For example, for the query `vacancy` relevant results include the term `vacant` as well. More such pairs are `discharge-surcharge`, `clause-subclause`, `constitution-constitute`, and `Vice President-President`.

Table 5.3: Precision for queries in *Constitution of India* collection. Unseen queries during training are emphasised in the table header.

	constitution	India	right	***vacancy***	***rajpramukh***	***deputy***	***discharge***
$P_{k=1}$	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$P_{k=10}$	1.00	1.00	1.00	0.60	0.80	1.00	1.00
$P_{k=25}$	1.00	0.96	0.92	0.24	0.76	1.00	0.88

Table 5.4: Precision for queries in *Bombay High Court records* collection. Unseen queries during training time are emphasised in the table header.

	Bombay	court	August	***judge***	***defendant***	***registrar***	***plaintiff***
$P_{k=1}$	1.00	1.00	1.00	1.00	1.00	1.00	1.00
FAISS $P_{k=10}$	0.90	1.00	1.00	1.00	1.00	0.90	1.00
$P_{k=25}$	0.88	1.00	0.88	1.00	1.00	0.92	1.00

Table 5.5: Precision for queries in *Tagore's papers*. Unseen queries during training time are marked in blue color.

	heart	light	নেই	sudden	thy	ছিল	অনুবাদ
$P_{k=1}$	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$P_{k=10}$	1.00	1.00	0.70	0.60	0.90	0.40	1.00
$P_{k=25}$	1.00	0.92	0.52	0.24	0.88	0.16	0.64

We also show the qualitative results obtained for the four collections in different settings. Figures 5.4 to 5.6 shows the top retrieval results for both seen and unseen queries. In Fig. 5.4 for the queries `judge` and `freedom`, we observe drastic changes in writing styles. Moreover, our framework retrieves accurate results despite the variations. This points out that the learnt embeddings for handwritten text are invariant to various handwriting styles. Another piece of evidence to support this claim is the positive results shown for 3 English handwriting styles in both Fig. 5.4 and Fig. 5.5. For the query `registrar`, the retrieval results contain both printed text images and handwritten images. It shows the robustness of these learnt embeddings. The retrieval results are promising without any prior training for these specific handwriting styles.

Figures 5.5 and 5.6 show that the incorrect retrievals match closely to the query keyword. We also show promising results in QbE retrieval mode for the Malayalam collection in Fig. 5.6(b). Despite the watermarks present in the images, the obtained results are accurate and relevant. QbE mode of querying is especially beneficial while searching for signatures and unknown symbols encountered in a collection.

Table 5.6: Precision for queries in *Mohanlal writings*. Unseen queries during training time are marked in blue color.

	സിനിമ	ജീവിതം	കുടുംബ	മോഹൻലാൽ	നടൻ	ഇന്ത്യ
$P_{k=1}$	1.00	1.00	1.00	1.00	1.00	1.00
$P_{k=10}$	1.00	1.00	0.50	1.00	0.70	1.00
$P_{k=25}$	1.00	1.00	0.20	0.44	0.28	0.72

(a) Bombay High Court records

(b) Constitution of India manuscripts

Fig. 5.4: Top-10 qualitative search results from collections: BHC records and CoI manuscript. Queries are shown at the top and incorrect retrievals are highlighted in red. Querying is done in QbS setting.

5.4.4 Comparative Results for Retrieval

This section discusses the time complexity and memory requirements of the indexing and retrieval module. We compare our index matching algorithm to two other approaches using K-D Tree and the nearest neighbour approach. The nearest neighbour approach involves an exhaustive search over all the samples in a given index. K-D Tree and IVF based search are both non-exhaustive. Search operations are conducted in selective lists to obtain retrieval results. We compare and report the retrieval time and the mAP obtained for these three index matching approaches in Table 5.7. We also report the index sizes (disk memory) for these methods. The retrieval time and index size are obtained by indexing and querying the fairly large BHC records collection. As mAP evaluation metric requires ground truth, we use the IAM test set to report this metric for search methods. The search method used in this work is both efficient in terms of retrieval time and QbS mAP. The retrieval algorithm used in this work decreases

(a) Top-5 matches

(b) Rare OOV query results

Fig. 5.5: Qualitative search results from Tagore's papers collection for QbS setting. (a)Top-5 results for English and Bengali queries. OOV queries are highlighted in blue and incorrect retrievals are highlighted in red. (b) Showing top-1 result for a few rare OOV words from the collection.

Table 5.7: Retrieval time vs. Accuracy for query matching algorithms. Search operations performed in an index with 200K samples with feature dimension as 2048.

Search method	Exhaustive	Retrieval Time (sec/query)	Index Size (GB)	QbS mAP (IAM test)
K-D Tree	No	0.59	3.5	0.95
Nearest Neighbour*	Yes	0.15	1.6	0.96
This work*	No	**0.02**	1.6	0.95

*Note: * Computed using* FAISS *library*

the time by 86% compared to the simple nearest neighbour approach. The choice of retrieval algorithm is crucial when the indexes are massive. Therefore, we also study the effect of index sizes on the search time. We observe that time required to perform brute-force search increases rapidly with an increase in index size compared to the inverted index based approach. Therefore, we use the inverted index based approach for searching.

(a) QbS Retrieval (b) QbE Retrieval

Fig. 5.6: Qualitative search results from Mohanlal's writings collection for QbS and QbE setting. Top row shows the queries. OOV queries are highlighted in blue and incorrect retrievals are highlighted in red.

5.5 Conclusion

Document retrieval from complex handwritten records containing an unseen vocabulary set is challenging. In this work, we discuss a document retrieval framework for unseen and unexplored collections. This framework can perform search operations on these collections without any fine-tuning for a specific collection. We discuss and evaluate our method both quantitatively and qualitatively on four different collections written in three scripts, of which two are Indic. Despite the availability of massive document collections, retrieval frameworks for Indian languages are barely studied Through this work, we present and discuss document retrieval in Indic collections. Our framework performs reasonably well on these sizeable collections. Even for rare words, the top results are accurate. Utilising this simplified framework we plan to introduce more Indic collections written in other Indic scripts. Finally, we also present a demonstration to display the usefulness of the retrieval framework.

Chapter 6

Conclusion and Future Directions

The capability to recognize and search across handwritten text is fundamental to any modern document analysis systems. There is an immediate need to provide content-level access to the millions of manuscripts and personal journals, large court proceedings and also develop HTR applications to automate processing of medical transcripts, handwritten assessments etc. This thesis discusses the problem of handwritten text recognition and handwritten search for Latin and Indic scripts. Firstly, we defined the context, formulated the tasks and discussed the associated challenges in chapters 1 and 2. We also discussed seminal and related approaches in chapter 2 along with analysis of publicly available handwritten datasets.

Chapter 3 focuses on improving an existing text recognition pipeline. To this end, we discuss the basic HTR pipeline in this chapter and propose a novel training setup to further improve the HTR architecture. The suggested approach introduces a multi-head training approach, which involved training two heads of the same HTR architecture with shared weights. We successfully apply the proposed approach on four benchmark datasets in Latin and Indic scripts. We report an average 18% decrease in character error rates across the benchmark datasets. Our training approach is a simple and effective. It generates robust feature maps needed for text recognition. The proposed setup is flexible too, as it can be applied HTR architectures for further finetuning. The effectiveness of our method suggests that the current HTR architectures can be finetuned to further reduce error rates and develop robust HTR.

In chapter 4, we focus majorly on developing HTR for Indic scripts. Firstly, we introduce IIIT-HW-WORDS dataset, a collective handwritten dataset for 10 prominent Indian scripts. We hope the scale and the diversity of IIIT-INDIC-HW-WORDS dataset in all 10 prominent Indic scripts will encourage research on enhancing and building robust HTRs for Indian languages. For the introduced dataset, we establish a high baseline for text recognition and discuss the effectiveness of the HTR modules. We also conduct a brief study to explore the utility of pre-training recognizers on other scripts. Future direction for developing Indic-HTR should include enriching the dataset with more writers and natural variations. Another important direction is to create handwritten data at line, paragraph and page level. Generating synthetic handwritten datasets is another promising area that is worth exploring for Indic HTR. We believe it is essential to continue improving Indic datasets to enable Indic HTR development.

Chapter 5 discusses an end-to-end document retrieval framework for unseen and unexplored collections. Document retrieval from handwritten records containing an unseen vocabulary set is challenging as it requires to search and spot keywords from an open vocabulary. Retrieval frameworks for Indian languages are barely studied despite the availability of massive data. Through this work, we present and discuss document retrieval in Indic collections. We propose an embedding based framework for querying and retrieving relevant information from handwritten collections. The proposed framework can perform search operations on these collections without any fine-tuning for a specific collection. We discuss and evaluate our method both quantitatively and qualitatively on four different collections written in three scripts, of which two are Indic. Our framework performs reasonably well on these sizeable collections. Additionally, we also present a demonstration to display the usefulness of the retrieval framework.

Related Publications

Conference Publications

- Santhoshini Gongidi, and C. V. Jawahar - **IIIT-HW-INDIC-WORDS : A Dataset for Indic Handwritten Text Recognition** , *International Conference on Document Analysis and Recognition (ICDAR), 2021*
- Santhoshini Gongidi and C.V. Jawahar - **Handwritten Text Retrieval from Unlabeled Collections**, *6th IAPR International Conference on Computer Vision & Image Processing (CVIP), 2021*

Bibliography

[1] R. Ahmed, W. G. Al-Khatib, and S. Mahmoud. A survey on handwritten documents word spotting. *IJMIR*, 2017. 9

[2] A. Alaei, U. Pal, and P. Nagabhushan. Dataset and ground truth for handwritten text in four different scripts. *IJPRAI*, 2012. 13, 14, 27

[3] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *TPAMI*, 2014. 39, 42

[4] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *ICCV*, 2019. 16, 20, 32

[5] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. In *CVPR*, 2019. 43, 45

[6] S. Bhowmik, S. Malakar, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri. Off-line bangla handwritten word recognition: a holistic approach. *Neural Computing and Applications*, 2019. 13, 14, 27

[7] T. Bluche, S. Hamel, C. Kermorvant, J. Puigcerver, D. Stutzmann, A. H. Toselli, and E. Vidal. Preparatory kws experiments for large-scale indexing of a vast medieval manuscript collection in the himanis project. In *ICDAR*, 2017. 11

[8] T. Bluche and R. Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *ICDAR*, 2017. 7, 16, 25, 36

[9] T. Bluche, H. Ney, and C. Kermorvant. Feature extraction with convolutional neural networks for handwritten word recognition. In *ICDAR*, 2013. 25

[10] R. M. Bozinovic and S. N. Srihari. Off-line cursive script word recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 1989. 6

[11] S. Chevalier, E. Geoffrois, F. Prêteux, and M. Lemaitre. A generic 2d approach of handwriting recognition. In *ICDAR*, 2005. 7

[12] M. Côté, E. Lecolinet, M. Cheriet, and C. Y. Suen. Automatic reading of cursive scripts using a reading model and perceptual concepts. *IJDAR*, 1998. 6, 7

[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 34

[14] K. Dutta. *Handwritten Word Recognition for Indic & Latin scripts using Deep CNN-RNN Hybrid Networks*. PhD thesis, International Institute of Information Technology Hyderabad, 2019. 12, 23, 27

[15] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar. Offline handwriting recognition on devanagari using a new benchmark dataset. In *DAS*, 2018. x, 14, 26, 27, 28, 32, 38

[16] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar. Towards spotting and recognition of handwritten words in indic scripts. In *ICFHR*, 2018. x, 14, 26, 27, 28, 29, 36, 38

[17] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar. Improving cnn-rnn hybrid networks for handwriting recognition. In *ICFHR*, 2018. 16, 20, 21, 32

[18] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character hmms. *PR*, 2012. 9, 11, 12, 25

[19] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern recognition letters*, 2012. 12, 13

[20] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou. A survey of document image word spotting techniques. *PR*, 2017. 9

[21] D. Goldhahn, T. Eckart, and U. Quasthoff. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, 2012. 29

[22] S. Gongidi and C. V. Jawahar. Iiit-indic-hw-words: A dataset for indic handwritten text recognition. In *ICDAR*, 2021. 45

[23] A. Granet, E. Morin, H. Mouchère, S. Quiniou, and C. Viard-Gaudin. Transfer learning for handwriting recognition on historical documents. In *ICPRAM*, 2018. 34

[24] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006. 7, 20, 32

[25] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems*, 21, 2008. 7

[26] E. Grosicki, M. Carré, J.-M. Brodin, and E. Geoffrois. Results of the rimes evaluation campaign for handwritten mail processing. In *ICDAR*, 2009. 12

[27] E. Grosicki and H. El-Abed. Icdar 2011-french handwriting recognition competition. In *ICDAR*, 2011. 25

[28] D. Guillevic and C. Y. Suen. Recognition of legal amounts on bank cheques. *Pattern Analysis and Applications*, 1998. 6, 7

[29] K. He, R. Girshick, and P. Dollár. Rethinking imagenet pre-training. In *CVPR*, 2019. 36

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 18, 25, 32

[31] J. J. Hull. Document image skew detection: Survey and annotated bibliography. In *Document Analysis Systems II*. World Scientific, 1998. 6

[32] R. Ishida. An introduction to indic scripts. In *Proceedings of the 22nd Int. Unicode Conference*, 2002. 31

[33] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 17

[34] J. C. A. Jaramillo, J. J. Murillo-Fuentes, and P. M. Olmos. Boosting handwriting text recognition in small databases with transfer learning. In *ICFHR*, 2018. 34

[35] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal. Database development and recognition of handwritten devanagari legal amount words. In *ICDAR*, 2011. 13, 14, 27

[36] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 2019. 46

[37] A. Kaltenmeier, T. Caesar, J. M. Gloger, and E. Mandler. Sophisticated topology of hidden markov models for cursive script recognition. In *ICDAR*, 1993. 7

[38] D. Kass and E. Vats. Attentionhtr: Handwritten text recognition based on attention encoder-decoder networks. In *DAS*, 2022. 8

[39] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997. 6, 7

[40] F. Kleber, S. Fiel, M. Diem, and R. Sablatnig. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In *ICDAR*, 2013. 12, 13

[41] S. Knerr, E. Augustin, O. Baret, and D. Price. Hidden markov model based word recognition and its application to legal amount reading on french checks. *Computer Vision and Image Understanding*, 1998. 6, 7

[42] M. Kozielski, P. Doetsch, H. Ney, et al. Improvements in rwth's system for off-line handwriting recognition. In *ICDAR*, 2013. 6

[43] P. Krishnan, K. Dutta, and C. Jawahar. Word spotting and recognition using deep embedding. In *DAS*, 2018. 42

[44] P. Krishnan and C. Jawahar. Generating synthetic data for text recognition. *arXiv preprint arXiv:1608.04224*, 2016. 45

[45] P. Krishnan and C. Jawahar. Hwnet v2: An efficient word image representation for handwritten documents. *IJDAR*, 2019. 9, 12, 39, 40, 42

[46] R. Kunwar, P. Mohan, K. Shashikiran, and A. Ramakrishnan. Unrestricted kannada online handwritten akshara recognition using sdtw. In *SPCOM*, 2010. 32

[47] A. W. Lee, J. Chung, and M. Lee. Gnhk: A dataset for english handwriting in the wild. In *International Conference on Document Analysis and Recognition*, pages 399–412. Springer, 2021. 12

[48] M. Lemaıtre, E. Grosicki, E. Geoffrois, and F. Prêteux. Off-line handwritten word recognition based on a bidimensional markovian approach with a large vocabulary. In *ICFHR*, 2008. 7

[49] U.-V. Marti and H. Bunke. Handwritten sentence recognition. In *ICPR*, 2000. 6

[50] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *IJDAR*, 2002. 3, 9, 11, 12, 25, 45

[51] J. Michael, R. Labahn, T. Grüning, and J. Zöllner. Evaluating sequence-to-sequence models for handwritten text recognition. In *ICDAR*, 2019. 8

[52] P. Natarajan, Z. Lu, R. Schwartz, I. Bazzi, and J. Makhoul. Multilingual machine printed ocr. *International Journal of Pattern Recognition and Artificial Intelligence*, 2001. 6

[53] E. Noya-García, A. H. Toselli, and E. Vidal. Simple and effective multi-word query spotting in handwritten text images. In *IbPRIA*, 2017. 11

[54] N. Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 1979. 6

[55] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *ICFHR*, 2014. 7

[56] T. Plötz and G. A. Fink. Markov models for offline handwriting recognition: a survey. *IJDAR*, 2009. 7

[57] D. M. Powers. Applications and explanations of zipf's law. In *New Methods in Language Processing and Computational Natural Language Learning*, 1998. 30

[58] J. Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *ICDAR*, 2017. 7, 16, 20, 25, 32

[59] T. M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *ICDAR*, 2003. 9

[60] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *CVPR*, 2003. 9

[61] T. M. Rath and R. Manmatha. Word spotting for historical documents. *IJDAR*, 2007. 45

[62] T. M. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *SIGIR*, 2004. 10

[63] P. P. Roy, A. K. Bhunia, A. Das, P. Dey, and U. Pal. Hmm-based indic handwritten word recognition using zone segmentation. *Pattern recognition*, 2016. 13, 14, 27

[64] M. W. Sagheer, C. L. He, N. Nobile, and C. Y. Suen. A new large urdu database for off-line handwriting recognition. In *ICIAP*, 2009. 14, 27

[65] J. A. Sánchez, V. Romero, A. H. Toselli, and E. Vidal. Icfhr2014 competition on handwritten text recognition on transcriptorium datasets (htrts). In *ICFHR*, 2014. 12

[66] J. A. Sanchez, V. Romero, A. H. Toselli, M. Villegas, and E. Vidal. Icdar2017 competition on handwritten text recognition on the read dataset. In *ICDAR*, 2017. 12

[67] J. A. Sanchez, A. H. Toselli, V. Romero, and E. Vidal. Icdar 2015 competition htrts: Handwritten text recognition on the transcriptorium dataset. In *ICDAR*, 2015. 12

[68] A. W. Senior and A. J. Robinson. An off-line cursive handwriting recognition system. *IEEE transactions on pattern analysis and machine intelligence*, 1998. 6

[69] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *PAMI*, 2016. 7, 16, 20, 32

[70] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 25, 32

[71] T. Strauß, G. Leifert, R. Labahn, T. Hodel, and G. Mühlberger. Icfhr2018 competition on automated text recognition on a read dataset. In *ICFHR*, 2018. 12

[72] S. Sudholt and G. A. Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *ICFHR*, 2016. x, 9, 10, 39, 40

[73] S. Thadchanamoorthy, N. Kodikara, H. Premaretne, U. Pal, and F. Kimura. Tamil handwritten city name database development and recognition for postal automation. In *ICDAR*, 2013. 13, 14, 27

[74] A. H. Toselli, V. Romero, J. A. Sánchez, and E. Vidal. Making two vast historical manuscript collections searchable and extracting meaningful textual features through large-scale probabilistic indexing. In *ICDAR*, 2019. 11

[75] A. H. Toselli and E. Vidal. Handwritten text recognition results on the bentham collection with improved classical n-gram-hmm methods. In *Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing*, HIP '15, pages 15–22, New York, NY, USA, 2015. ACM. 16

[76] E. Vidal, V. Romero, A. H. Toselli, J. Sánchez, V. Bosch, L. Quirós, J. M. Benedí, J. R. Prieto, M. Pastor, F. Casacuberta, et al. The carabela project and manuscript collection: Large-scale probabilistic indexing and content-based classification. In *ICFHR*, 2020. 11

[77] P. Voigtlaender, P. Doetsch, and H. Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *ICFHR*, 2016. 7

[78] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen. Data augmentation for recognition of handwritten words and lines using a cnn-lstm network. In *ICDAR*, 2017. 21

[79] T. Wilkinson and A. Brun. Semantic and verbatim word spotting using deep neural networks. In *ICFHR*, 2016. x, 9, 10

[80] T. Wilkinson, J. Lindström, and A. Brun. Neural word search in historical manuscript collections. *ArXiv*, 2018. 10, 11, 40, 47

[81] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *NIPS*, 2014. 34

[82] M. Yousef and T. E. Bishop. Origaminet: Weakly-supervised, segmentation-free, one-step, full page text recognition by learning to unfold. In *CVPR*, 2020. 13