Handwritten Word Recognition for Indic & Latin scripts using Deep CNN-RNN Hybrid Networks

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

Master of Science
in
Computer Science and Engineering
by Research

by

Kartik Dutta
201302018
kartik.dutta@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2019

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Handwritten Word Recognition for Indic & Latin scripts using Deep CNN-RNN Hybrid Networks" by Kartik Dutta, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. C.V. Jawahar

To

My Parents &

Sabaton

Acknowledgements

The successful completion of this thesis has been possible through the assistance of many people. I am forever indebted to all the people who have helped me.

First of all, I would like to thank Prof. C.V. Jawahar for introducing me to research. I'll forever remember him for being the first person who made me feel worthwhile as both a researcher and a programmer.

Words cannot express the aid that Minesh Mathew and Praveen Krishnan have lent to me whenever I was stuck or at a deadlock. This work would have been far more challenging without their support.

I would also like to extend regards to all the other friends at IIIT for their help and support at various stages of my work. I am also thankful to Silar Shaik, Varun Bhargawan, Siva Kumar and Mahender Reddy for helping with the annotation of datasets, server handling issues and scheduling various meetings to fast track my work.

Lastly, I would like to thank my family for showering me with their love and financial support.

– Kartik Dutta

Abstract

Handwriting recognition (HWR) in Indic scripts is a challenging problem due to the inherent subtleties in the scripts, cursive nature of the handwriting and similar shape of the characters. Though much research has been done in the field of text recognition, the focus of the vision community has been primarily on English. Furthermore, a lack of publicly available handwriting datasets in Indic scripts has also affected the development of handwritten word recognizers and made direct comparisons across different methods an impossible task in the field. Also, due to this lack of annotated data, it becomes challenging to train deep neural networks which contain millions of parameters. These facts are quite surprising as there are over 400 million Devanagari speakers in India alone.

We first tackle the problem of lack of annotated data using various approaches. We describe a framework for annotating large scale of handwritten word images without the need for manual segmentation and label re-alignment. Two new word level handwritten datasets for Telugu and Devanagari are released which were created using the above-mentioned framework. We synthesize synthetic datasets containing millions of realistic images with a large vocabulary for pre-training using publicly available Unicode fonts. Later on, pre-training using data from Latin script is also shown to be useful to overcome the shortage of data.

Capitalizing on the success of the CNN-RNN Hybrid architecture, we propose various improvements in its architecture and training pipeline to make it even more robust for handwriting recognition. We now change the network to use a Resnet-18 like structure for the convolutional part along with adding a spatial transformer network layer. We also use an extensive data augmentation scheme involving multi-scale, elastic, affine and test time distortion.

We outperform the previous state-of-the-art methods on existing benchmark datasets for both Latin and Indic scripts by quite some margin. We perform an ablation study to empirically show how the various changes we made to the original CNN-RNN Hybrid network have improved its performance with respect to handwriting recognition. We dive deeper into the working of our networks convolutional layers and verify the robustness of convolutional-features through layer visualizations. We hope the release of the two datasets mentioned in this work along with the architecture and training techniques that we have used instill interest among fellow researchers of the field.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The creation and dissemination of handwritten documents remains pervasive for humans as a personal choice of communication other than speech. Understanding text written in handwritten documents also enables access to the vast information present in handwritten content in numerous domains where either the digital technologies are yet to be fully adopted (say court proceedings) and the places where the medium of handwriting is still pervasive (e.g., personal notes, content managed in schools and educational institutions, etc.).

Handwritten text recognition has been a popular research area for many years due to various applications such as making handwritten manuscripts digitally searchable [8], postal automation [5, 9, 10] (refer to Figure 1.1), digitizing handwritten forms [11] such as the form shown in Figure 1.2, etc. Content level access of scanned handwritten documents written in native scripts has a profound value of interest for both historians and the local people. There are numerous handwritten manuscripts [6], notes and essays [12] with high literary content which have been scanned as part of digital archives [13] but are not searchable.

With there being 22 official languages in India and many more used for communication it is imperative that more research is done into both improving HWR accuracy as well as developing end to end systems to be used for solving the problems mentioned above. Solving these problems is an integral part of Digital India [14] initiative and in that regard, the Government of India has even opened a specific project looking into HWR for Indic scripts [15].

Formally, a handwritten word recognition (HWR) problem is formulated into two parts, where given a word or line image, the task of the recognizer is to predict the character string ($\mathbf{w}$) which is later given to a linguistic engine which constrains $\mathbf{w}$ to form a valid word. These constraints can be either from lexicon $\mathcal{L}$ or a language model $P(\mathbf{w})$. Although there are methods which do not apply linguistic constraints, they are relatively worse in performance and cannot reliably be given to end applications.

Most of the research in HWR has focused on Latin scripts, with very few works in the space of Indic scripts. Most of the research in Indic scripts has instead centered around printed text recognition

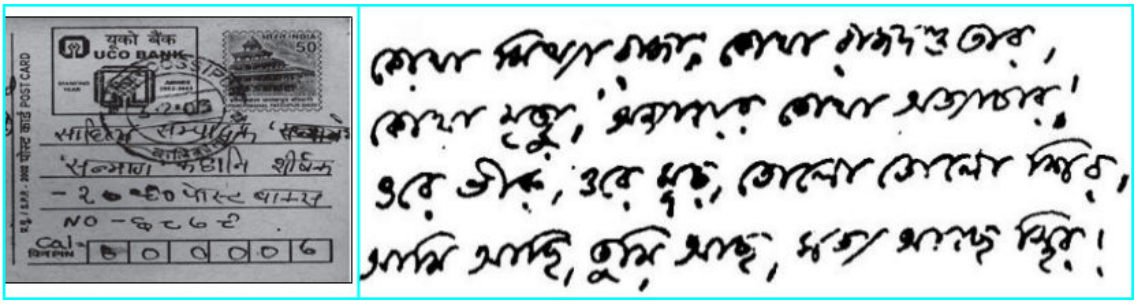

Figure 1.1: Two examples of the applications of Indic script based Handwriting Recognition. By recognizing handwritten PIN codes, city names, etc. [4, 5], there is much scope of postal automation (left box). The right box shows an example of Bangla writing from a manuscript of Rabindranath Tagore. Handwriting Recognition can make such digitized manuscripts searchable and indexable, making access easier for other scholars working on Indian manuscripts.

(OCR) [16, 17, 18]. There are various reasons why Indic script HWR research is lagging. One is the inherent complexity of Indic scripts such as having a much larger number of characters than Latin scripts. Another reason is that there is a severe lack of publicly available datasets annotated at the word or line level. There are some scripts like Malayalam that to our knowledge has no publicly available handwriting dataset. The currently available datasets also suffer from various issues. Compared to any standard Latin off-line handwriting recognition dataset like IAM [19], Indic scripts datasets are much smaller in size, both in terms of the number of words and writers. Also, due to the lack of standard benchmark datasets for any Indic scripts, it is hard to compare different methods directly.

In recent times, there have been tremendous improvements in recognition rates [20, 21, 22] owing to the success of underlying machine learning models using deep neural networks. The improvement in performance can be credited to: (i) better architectures such as convolutional neural networks (CNN), recurrent neural networks (RNN), (ii) better learning schemes and regularizers, (iii) the availability of large scale of annotated data, and (iv) increased computational capacity using GPUs. Traditionally, in the domain of text recognition, HMMs [23] or RNNs (especially BLSTMs [24] or MDLSTMs [25]) along with CTC loss [26] have been popularly used due to its inherent ability to process temporal sequences. More recently, it was shown that a hybrid scheme of using convolutional recurrent architecture [27] where the convolutional layers are meant for feature extraction, with the features being subsequently passed on to a BLSTM network along with CTC loss, works better than other schemes.

In this thesis, we specifically try to solve the problem of offline handwriting recognition with pre-segmented words provided to us. To overcome the problem of lack of annotated data we use a multi-pronged strategy. We release two new benchmark datasets for Indic script HWR. However, since collecting actual samples is an expensive task and deep learning solutions are data hungry, we also use an extensive pre-training and data augmentation pipeline. We make improvements in the architecture that

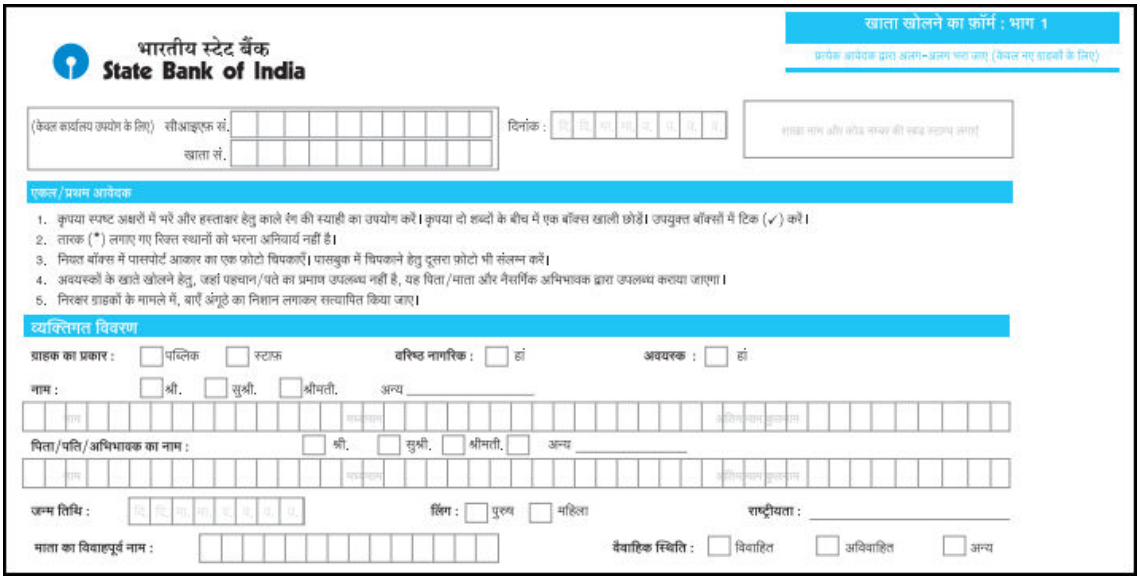

Figure 1.2: Another example of the application of Handwriting Recognition in Indian setting: Bulk processing of highly structured forms which are filled by humans.

was introduced by [27] by adding state of the art deep learning components such as spatial transformer networks [28] that are helpful in increasing HWR performance. We present state of the art results (with and without lexicon based decoding) for both Indic script and Latin based datasets, showing that the methods discussed in this work are language agnostic.

1.1 Contributions

In this thesis, we try to solve the problem of HWR for Indic scripts by releasing new datasets and making improvements to the CNN-RNN Hybrid network. The significant contributions of this thesis are listed below:

- **Public Benchmark Datasets:** To further facilitate research in this field, we make two benchmarking datasets available; IIIT-HW-DEV dataset and the IIIT-HW-TELUGU dataset for the problems of Devanagari and Telugu HWR, respectively. To the best of our knowledge, these are the only datasets with Unicode labelling and having over 100k annotated words that have been publicly released for Devanagari and Telugu.
- **An Efficient Dataset Collection Method:** We mention a new dataset collection method for HWR (both words and lines) with a negligible requirement for human verification of the segmentation. Usual methods for creating word level datasets have to perform segmentation at both line and word level and require manual alignments of words with labels. Our method does segmentation and label assignment automatically without any need of alignment.

- **Novel Pre-training Pipeline:** We propose a novel pre-training and data augmentation pipeline. Firstly we pre-train our models through handwritten data from an unrelated script, having relatively more annotated real data, such as Latin. Secondly, we use a synthetic data generation pipeline, inspired from [29] to create vast amounts of images which resemble real-world handwritten images and span a large vocabulary set. While training our models on the actual data we use multiple data augmentation techniques such as elastic and multi-scale distortion. For Indic script based languages, getting large quantities of annotated data is often difficult and hence such a large-scale pre-training and data augmentation pipeline tremendously helps in improving the recognition accuracy.

- **Using a Deep CNN-RNN Hybrid Network:** We are one of the first works in the field of HWR which uses a Resnet-18 [20] structure for our convolutional layers in the CNN-RNN hybrid network along with using a spatial transformer network [28].

- **Ablation Studies:** We perform ablation studies on two Indic script datasets, IIIT-HW-DEV and IIIT-HW-TELUGU showcasing how all the primary components of our HWR solution such as data augmentation, synthetic data pre-training, spatial transformer network, etc. helped in making a robust model.

- **Establish new *state-of-the-art* accuracy:** We beat the current state-of-the-art (SOTA) solutions for both popular Indic script datasets (ROYDB [1]) and Latin datasets (IAM [19], RIMES [30], GW [31]). We present results in both lexica based and free decoding settings which beat the current benchmarks by a large margin.

- **Insights into Convolutional layers:** We provide insights by creating layer visualizations into the workings of the convolutional layers in our Deep CNN-RNN Hybrid network and thus verify it's robustness over the traditional methods of using hand-crafted features.

1.2 Thesis Layout

The organization of this thesis is as follows. Chapter 2 gives an overview of the multiple languages and scripts currently in use in India. We talk about the current Unicode encoding standard used for representing Indic scripts. We also talk about how we chose our basic unit of recognition. Finally, we talk about the challenges that make HWR for Indic scripts more complicated as compared to Latin.

Chapter 3 talks about the present lack of HWR datasets for Indic scripts. It also talks about how we collected data for the IIIT-HW-DEV and IIIT-HW-TELUGU datasets and how our methodology differs from the conventional way HWR datasets are created. We then go over some statistics regarding both the datasets that were created as part of this thesis.

We start with a detailed literate review in chapter 4. We talk about the various pre-processing, segmentation, feature extraction and recognition methods that various works in Indic script HWR use. We then describe the primary building blocks of the solution architecture such as the Spatial Transformer Network, Resnet modules, etc. Our training pipeline which involves pre-training with synthetic data and real data from Latin script is described next. Finally, we talk about the various data augmentation strategies used in this work.

Chapters 5 and 6 present the results of the network architecture presented in this thesis on various benchmark datasets for Indic scripts and Latin respectively. We discuss the synthetic datasets that are created using the synthetic rendering pipeline. Ablation studies are presented in chapter 5 to empirically validate the training and architectural decisions taken while finalizing our network. A brief comparison is made between the different benchmark datasets and we compare the error rates of the method presented in this work with other state of the art methods present in literature.

Chapter 7 concludes this thesis by consolidating all the contributions of this thesis. We also leave pointers and extensions for researchers working in this field to pursue.

Chapter 2

Indian Scripts: Overview

In this chapter, we give a high-level description of Indic scripts and the popular methods of encoding their textual representation. We also talk about the basic unit of representation that we use while conducting our recognition experiments in the later chapters.

2.1 Structure of Indian Languages

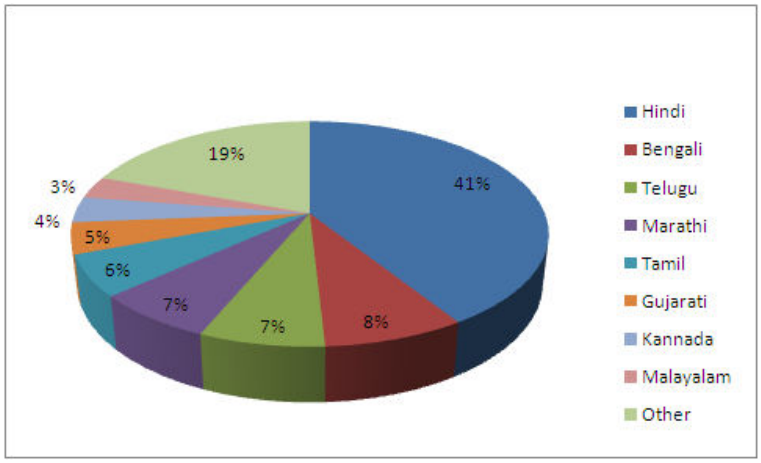

Figure 2.1: Percentage of native speakers of the eight largest scheduled languages in India, according to the 2001 Census of India

According to the 2001 census of India [32], there are 22 major languages in India. Hindi is the mother tongue for 41% of Indians, followed by Bengali (8%) and Telugu (7%). Figure 2.1 shows the percentage of native speakers of the top 8 scheduled languages in India, by the number of native speakers. Rest of the speakers of all the languages spoken in India make up only 19% of the total speakers.

Figure 2.2: Here we see how the word Bharat is written in different Indic scripts. The left box shows how it is written in Devanagari based languages while the right box shows how it is written in Dravidian languages.

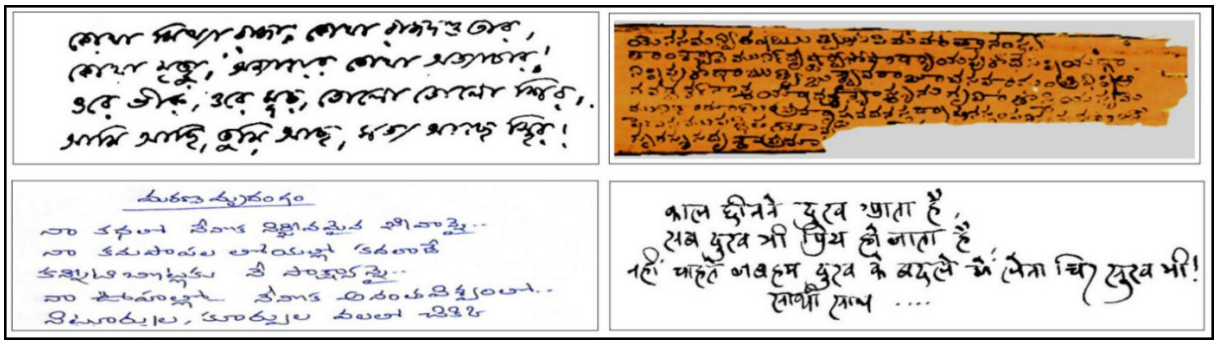

Figure 2.3: Clockwise from the left:(a) Excerpt from the Bangla poem "Namashkar" (1907) written by Rabindranath Tagore, (b) A Telugu palm leaf text [6], (c) An example of Devanagari writing from a Hindi poem written by Amitabh Bachchan and (d) Contemporary Telugu writing.

There are two dominant language families in India, namely Indo-Aryan and Dravidian language family. About 75% of the population are speakers of Indo-Aryan languages which includes Hindi, Bengali, Marathi, Urdu, Gujarati, Punjabi, and Assamese. Dravidian language family accounts for some 215 million speakers or approximately 20% of the population. This language family is predominant in the South of India, with Telugu, Tamil, Kannada and Malayalam being part of the Dravidian family. Figure 2.2 shows the word "Bharat" written in different Indic scripts. Figure 2.3 shows a few handwriting excerpts across Indic scripts.

2.2 Unicode Encoding Standard

Unicode encoding [33] is now the most widely used encoding standard for Indic scripts. It uses a 16 bit representation for each symbol and is designed to be a multilingual encoding that does not require any escape sequences or switching between scripts. Unicode assigns each language a separate block

Table 2.1: Unicode ranges of major Indian scripts

Language	Unicode Begin	Unicode End
Hindi	0900	097F
Bangla	0980	09FF
Gurmukhi	0A00	0A7F
Kannada	0C80	0CFF
Telugu	0C00	0C7F
Tamil	0B80	0BFF
Malayalam	0D00	0D7F

to represents its symbols. It has characters like the Zero Width Joiner (ZWJ), Zero Width Non Joiner (ZWNJ), Virama, etc. which allow Unicode fonts to render any character that is present in Indic scripts. Table 2.1 shows the Unicode range for the major Indian scripts. There are still works which use ASCII based transliteration [34] to represent the label of an image or try to use a custom encoding based on the ASCII system [1]. Such schemes should be avoided in use as they do not represent a significant number of the unique characters in Indic scripts.

2.3 Basic Unit of Recognition?

In Indian languages, the question of choosing the basic recognizable unit of a word is debatable. A single glyph can have multiple Unicode characters associated with it. We could choose to use Unicode characters as a base character of recognition or *aksharas* which are made up of multiple Unicode characters.

Indic scripts are syllabic and follow the abugida or alphasyllabary writing system [35]. The alphasyllabary system contrasts with the alphabet writing system used in English. In the alphabet system, vowels have status equal to consonants whereas, in an alphasyllabary system, consonant-vowel sequences are the basic unit of a word. So, while in English a word can begin with any of the alphabets, in Indic scripts this is not allowed. Each such consonant-vowel sequence unit has to start with on a consonant letter and the presence of a vowel is optional.

In Indic scripts this consonant-vowel sequence unit of is called an *akshara* or ortho-syllable [36] and is considered the basic writing unit in Indic scripts. Figure 2.4 shows a word in Devanagari being split into *aksharas*. Multiple Unicode characters can be used to represent an *akshara*, in contrast to English where a single Unicode character represents an alphabet. *Aksharas* are similar to syllables in English follows and follow the pattern $C * V$, where C is a consonant and V is a vowel.

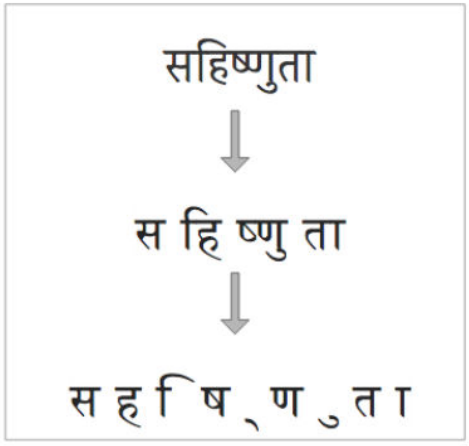

Figure 2.4: Here we see an example of *akshara* splitting for a word in Devanagari. The top image is the original image, the middle image shows it's *akshara* split, while the bottom-most image shows the Unicode level split. Clearly, an *akshara* can be made up of multiple Unicode characters.

As per [37] there were over 20,000 *aksharas* in a Telugu corpus of 39 million words, with 5000 of them occurring most frequently. Since most handwriting datasets have vocabulary sizes much smaller than these, if we set our basic unit of recognition as the *aksharas* and cover all 20,000 of the possible variations we would have very little or no training data for most of them. We could only cover the *aksharas* that are present in the vocabulary that we are using as was done in [36], but then our recognizer would not be able to recognize any *akshara* out of these set of characters. Also, creating *akshara's* out of Unicode characters requires knowledge about the particular Indic script we are trying to recognize.

Due to the reason of requiring a large vocabulary of words to have significant *akshara* coverage and the inability to create language agnostic solutions, we decided to use Unicode characters as our basic character of recognition in all the experiments we conduct in the following chapters. Also, many recent works like [38, 39, 40] didn't use the *akshara* as their basic unit of recognition.

2.4 Unique Challenges in HWR Design for Indic Script

Indic scripts have some inherent characteristic which adds new challenges over and above those faced in Latin HWR. For example in Devanagari and Bangla, the shape of a consonant is modified when a vowel follows it and such a character is referred to a *modified* character or modifier. In some instances, when a consonant follows one or more consonant(s), a character with a different shape gets formed which has an orthographic shape and is called a *compound* or *conjunct* character. Figure 2.5 shows a few examples of *compound* characters and *modifiers* for Bangla and Devanagari.

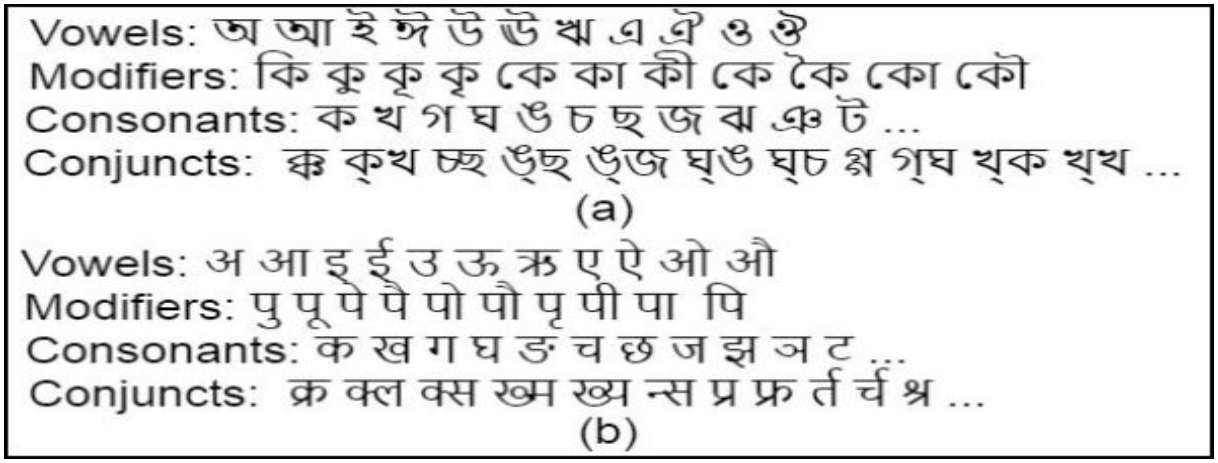

Figure 2.5: Few examples of vowels, modifiers, consonants and conjuncts in (a) Bangla and (b) Devanagari.

Figure 2.6: A few examples of pairs of cursive characters in Devanagari which can be easily confused with each other due to their similar looking shapes.

As per [41] there are over 360 *compound* characters in Devanagari with a non-negligible frequency of occurrence. Because of *modified* and *conjunct* characters, there are issues with segmentation of characters particular to in Indic scripts, making character segmentation based recognition solutions less viable for solving word level recognition problems.

There is also a problem of lack of publicly available annotated handwriting datasets for Indic scripts, for a multitude of reasons mentioned in depth in the next chapter. Many characters in Indic scripts are easily confusable with each other due to the cursive way of writing Indic scripts and the similarity between the shape of these characters. Figure 2.6 shows a few examples of such characters for Devanagari. Due to a large number of *conjunct* and *modified* characters, this issue is more pronounced for Indic scripts as compared to Latin script.

Chapter 3

Data Collection for Indic Scripts

In order to prevent over-fitting in deep neural network architectures requirement of large amounts of data is a must [42] since deep learning networks contains millions of parameters. If we also consider the fact that there are currently no standard benchmark datasets available for comparison of various works in Indic script HWR, we can see why there is an acute need for creating general purpose handwritten benchmark datasets for all the Indic scripts.

This chapter talks about the newly proposed technique of dataset collection for handwriting recognition and how its more efficient compared to the current dataset collection technique for handwriting recognition. We also mention details about the two datasets released as part of this work.

3.1 Need for datasets in Indic scripts HWR

Unlike Latin scripts, there is a severe lack of publicly available datasets for Indic scripts that are annotated at the word or line level. There are many datasets available which are isolated character level datasets, but those are not useful for word level recognition. There are some scripts like Malayalam, Gurmukhi that to our knowledge has no publicly available handwriting dataset at the word or line level. For the scripts where datasets are available, those datasets also suffer from various issues. Figure 3.1 shows a few sample word images from various publicly available Indic datasets.

Compared to any standard Latin off-line HWR dataset like IAM [19] Indic script datasets are much smaller in size, regarding both the number of words and writers, as can be seen in table 3.1. The IAM dataset had over 500 writers, while at max the CMATERDB2.1 dataset has around 300 writers. Most of the datasets contain only around 30k words, one-fourth of the number of words in the IAM dataset. Also, most of the datasets have a small vocabulary of at maximum 1000 word while IAM has a vocabulary of over 10k words. Such small and specific vocabulary would contain very little of the total possible number of unique *akshara's* that are possible for any Indic script and would lead to over-fitting.

Figure 3.1: A few sample images from the publicly available Indic script world handwriting datasets. The first row contains word images from LAW, the second row from RoyDB Devanagari track, the third row from RoyDB Bangla track and the last row from CMATERdb2.1 dataset.

Many of these datasets are not labeled using Unicode labels. Instead, datasets like LAW, CMATERdb2.1 use ASCII labeling by transliterating the words from the Indic script to English. Some datasets like RoyDB use custom ASCII based encoding which is specific to their datasets. Such variation in labelling style makes it harder to verify results. Also, Sabir et al. [43] empirically showed that LSTM's learn an implicit language model when trained on OCR data. Since Indic scripts have many more characters than their Latin counterparts (like Devanagari having over 400 unique characters compared to 52 in Latin), this makes it essential to have vast quantities of training data to avoid overfitting, notwithstanding the fact that deep neural networks contain millions of parameters.

Such a lack of large quantities of structured and publicly available data has been a significant cause for Indic scripts handwriting recognition community lagging behind it's Latin counterpart [48]. Most works reported results on small datasets created privately and never made available public [49]. This makes comparing performance across different methods impossible. Also, the accuracy numbers reported by most papers on text recognition is very high, even though the solutions may not scale in other settings. However in Latin, for over a decade all HWR works have been showing results using the IAM dataset [23, 24, 50, 51], making it trivial to compare results across different papers.

Most of the Indic script datasets were curated for a specific task like recognizing words from bank cheques (LAW dataset), Tamil city names (Tamil-DB), etc. For a fair comparison between HWR

Table 3.1: Public HWR datasets for Indic scripts, ignoring character datasets. Here GT Level refers to the modality at which labelling was done. The last two datasets were released as part of this thesis.

Name	Language	GT Level	#Writers	#Words	#Vocabulary
PBOK [44]	Bangla	Page	199	21K	925
	Oriya	Page	140	27K	1040
	Kannada	Page	57	29K	889
CMATERDB1.1 [45]	Bangla	Line	40	30K	567
CMATERDB2.1 [34]	Bangla	Word	300	18K	120
ROYDB [1]	Bangla	Word	60	17K	525
	Devanagari	Word	60	16K	1030
LAW [46]	Devanagari	Word	10	27K	220
TAMIL-DB [10]	Tamil	Word	50	25K	265
IIIT-HW-DEV [47]	Devanagari	Word	12	95K	11,030
IIIT-HW-TELUGU [3]	Telugu	Word	11	120K	12,945

systems, a standard benchmarking dataset is of utmost importance. Such a dataset would help the community to correctly rank all proposed solutions and establish a *state-of-the-art* solution.

3.2 Dataset Creation Methodology

In this section, we talk about the currently favored method of collecting data for handwriting recognition and its shortcomings. We also show how the newly proposed technique of dataset collection through QR code and individual boxes requires lesser human involvement in correcting segmentation errors. Our technique, although shown for words is just as applicable for lines or even paragraphs.

3.2.1 The default approach

The currently favored approach [19, 44] for collecting handwriting data is as follows. First, a corpus like the LOB corpus [52] is chosen. From within such a corpus, various paragraphs are chosen and printed onto forms. Human annotators are required to write the printed paragraph shown to them. Figure 3.2(a) shows such a handwritten form. Annotators are free to write using the instrument of their choice and are not constrained by space.

After the scanning of the handwritten forms is complete, algorithms are run to perform line and then word level segmentation. Interested readers can refer to [19] to understand what segmentation

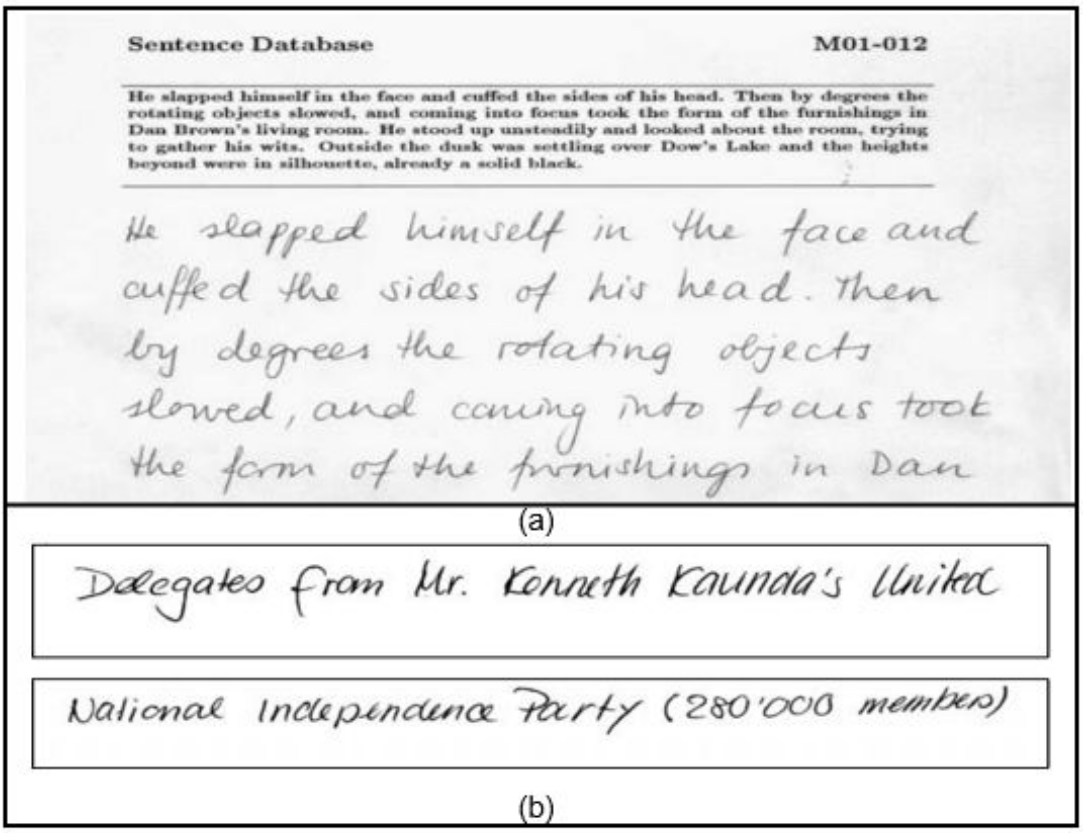

Figure 3.2: (a) Usually for collecting handwriting data, annotators are required to copy the text shown to them with little restriction on spacing. (b) First lines are segmented out and then individual words are segmented out, only then are we able to get a corpus annotated at the word level.

algorithms were used. Figure 3.2(b) shows a few segmented out handwritten lines. As per [19] a 541 line portion (subset) of the IAM dataset had a word segmentation accuracy of 94.92%. Also, roughly 14 hours were spent to ensure that the word labelling was correct (else realignment would be performed).

3.2.2 Approach used in this work

Since we were only creating a dataset for word level recognition, the above approach was suboptimal since we have to perform both line and word segmentation along with having to spend time manually correcting/verifying annotations.

In our approach, we first assembled a list of roughly 15K words, provided by [53] for Devanagari and Telugu. These words were chosen from random websites over the web for both the scripts. We pruned out words in the vocabulary with characters outside of the particular script's Unicode set. The

words present in the vocabulary were sampled in such a way that words were ranging from very high frequency to words with a very low frequency of occurrence.

We first randomly permuted our vocabulary of words to ensure that we maximize the variance across writers. We fixed a constraint that every word in the vocabulary would have 10 image samples for it in the dataset. However, in a general dataset the number of image samples of a word is dictated roughly by Zipf's law [54]. We created a mapping between each word in the vocabulary and a unique integer identifier (ID). We generated QR code for all the integers that were in the vocabulary identifier list.

We synthesized images with printed text containing the words in the vocabulary. Then we generated boxes as shown in figure 3.3 (d). In the middle of each box, there is a printed word image from the vocabulary, at the top is the QR code corresponding to the integer identifier for the word/image and space at the bottom for the annotator to write the sample. Each box had a fixed height but variable width (to maintain aspect ratio for the printed image), to account for the variable word length. Within each box, the image, QR code and sample had equal height space to make segmentation of these three components within a box a trivial task. Each box was padded with a certain amount of black pixels which would later help in the decoding process.

The process of creating forms from the prepared box images followed a simple algorithm. We tried to fit in as many boxes as possible in a row (with gaps between boxes and corners) and once the next box to be placed would overflow in a row, it was moved onto the next row until a page had no more space for the next box to be placed. Now, the current form is finalized and we started with creating a new blank form and start the above process again. We kept creating and filling forms till all the words in the vocabulary were present exactly once. We then created nine more copies of each form created to get ten samples for each word in the dataset.

Once a filled form was scanned, it was binarized using Otsu's [55] method (see figure 3.3 (a) and (b)). After binarization, we fill holes in the binary image. A hole is a set of background pixels that cannot be reached by applying the flood fill algorithm with a foreground seed in a background pixel present at the edge of the image [56]. Now as we can see from figure 3.3 (c), all the boxes are clearly marked as foreground and form disjoint connected components. Each of these boxes is then processed separately, as in figure 3.3 (d). Within a box, segmentation is easily performed and after decoding the QR code integer value, we store the image and label it through the mapping table with the ID obtained through the QR code.

Thus, by using this procedure, we were able to segment and label the written words automatically. However, we did not manually correct the segmentation. We manually performed an evaluation of our automatic segmentation procedure on a few filled forms. Here we obtain more than 99% word segmentation accuracy with our method as compared to 94.92% word segmentation accuracy reported

in IAM (for a small subset of their dataset). Also, we did not have to put in any manual labor for re-alignment, unlike the IAM dataset.

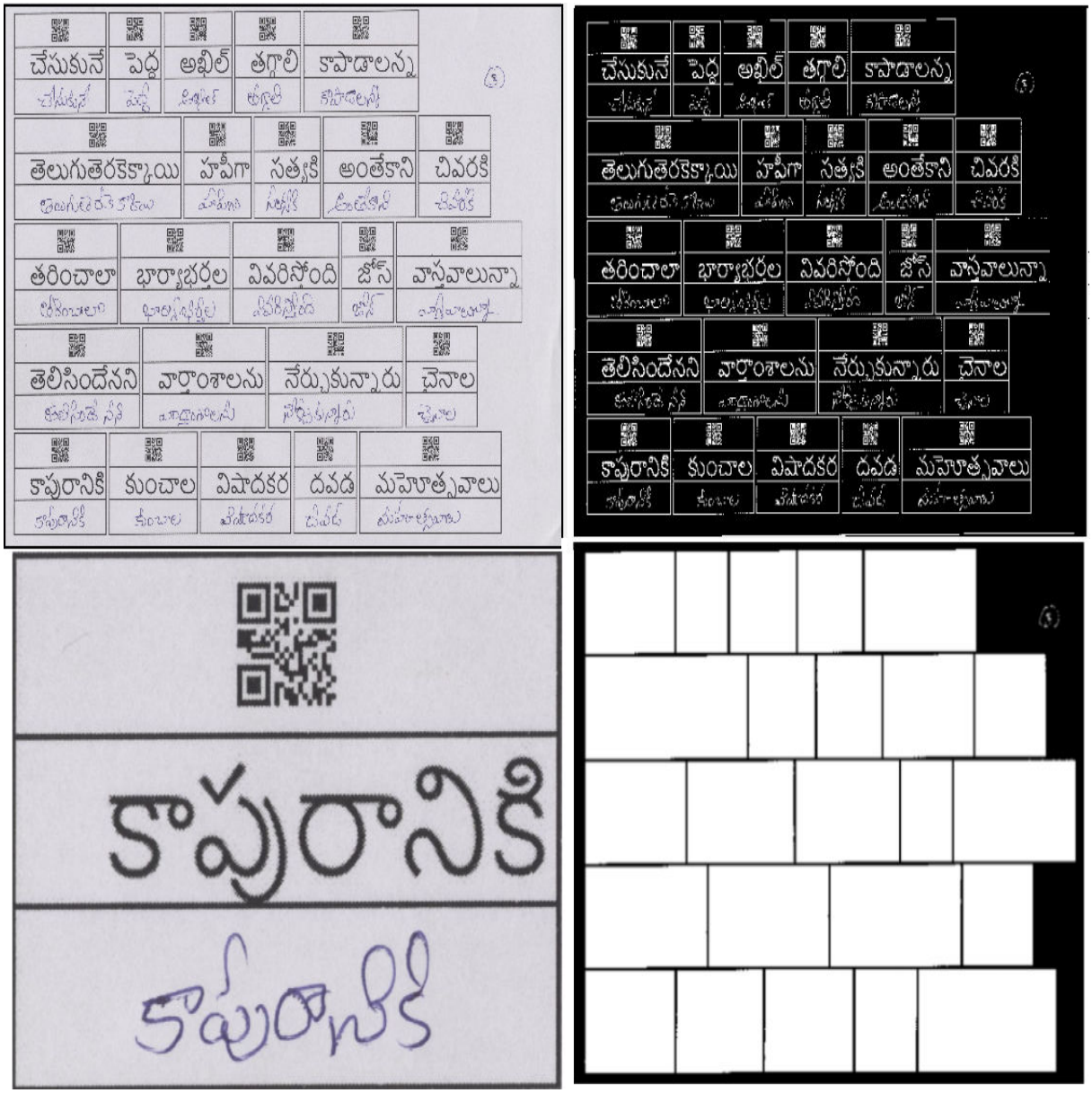

Figure 3.3: Clockwise from the left:(a) A filled form, (b) Form after applying binarization by Otsu's method, (c) Form after filling holes in the binary image. Afterwards, all of the white rectangular patches on the form are individually processed one by one, (d) An enlarged image of an extracted filled up box. In the images (b) and (c) the black color represents the background while the white color represented the foreground.

3.3 Created Dataset Details

In this section, we talk about the statistics and other particular details of the IIIT-HW-DEV and IIIT-HW-TELUGU datasets that we have released.

3.3.1 IIIT-HW-DEV

For the IIIT-HW-DEV dataset, we collected a total of 95,381 word samples from 12 different individuals with different educational backgrounds and ages. The writers were free to use pens of their choice and write the words in their natural style.

The train, validation and testing split was done such that there is no overlap of writers between the three sets. The training, validation and testing set are roughly in a ratio of 70:15:15 and contain nearly 70k, 12k & 12k annotated word images respectively. The dataset is annotated using Unicode. A vocabulary of 11,030 unique words was used in this dataset. On an average, each word consists of six basic Unicode characters.

3.3.2 IIIT-HW-TELUGU

For this dataset, we collected a total of 1,18,515 words samples from 11 different individuals with different ages and educational backgrounds. Here each writer has written nearly 10K words on an average. The writers were free to write the words in their natural style instead of copying the word image shown to them. They were allowed to use any pens of their interest.

The train, validation and test sets were created such that there is no overlap of writers between any of the three sets. The training, validation and testing set contains 80637, 19980 and 17898 annotated word images respectively, roughly in the ratio of 70:15:15. This dataset too is annotated using Unicode. A vocabulary of 12,945 unique words was used in this dataset. On an average, each word consists of nine basic Unicode characters. Figure 3.4 shows a few examples of segmented Telugu word images that are part of the IIIT-HW-TELUGU dataset.

Both of the datasets the forms were scanned using a HP flatbed scanner in a resolution of 600 DPI in color and stored using the JPEG format. They are available for download at:
https://cvit.iiit.ac.in/research/projects/cvit-projects/indic-hw-data

Figure 3.4: Sample word images for IIIT-HW-TELUGU dataset. The first two rows show different words that have been written by the same writer. The last two rows show examples of the same word that has been written by different writers. One can notice both the inter and intra class variability in style across the collected samples.

3.4 Summary

In this chapter, we talked about a new QR code based technique of dataset collection for handwriting recognition and how it is more efficient compared to the currently most popular technique with lesser segmentation errors and human involvement. We mentioned details about the IIIT-HW-DEV and IIIT-HW-TELUGU datasets released as part of this work. Both datasets contain almost 100k word samples, a first for Indic script datasets. We hope that these two datasets will be accepted as benchmark datasets by the research community.

Chapter 4

Methodology

4.1 Introduction

We can divide Handwriting recognition into either offline or online based. Offline handwriting recognition deals with recognizing handwritten content from scanned or photographed sources. In online HWR we also have the time series data which gives us the sequence of hand movements which led to the formation of the word to be recognized [57].

This chapter starts with a Literature survey regarding the various steps that are present in a HWR system, with particular emphasis for works related to Indic script HWR. We mostly restrict ourselves to offline recognition based works where pre-segmented word images are available since that is the problem setting used throughout this thesis.

We then move the discussion onto the significant components of the final solution architecture chosen for the recognition experiments performed as part of this thesis. We also discuss the network training and data augmentation pipeline that was used. Finally, we talk about a few implementation details regarding the architecture that we used.

4.2 Literature Survey

Handwriting recognition is generally a multi-stage process. Figure 4.1 illustrates the various steps in a typical Handwritten word recognition system. Each of these steps and the current techniques in the domain of Indic scripts is discussed in this section.

4.2.1 Pre-Processing

Pre-processing is a crucial step in HWR for Indic scripts [58]. This step is fundamental to improve the discriminating nature of the pixels or raw features being computed from input images. We can generally categorize these efforts into these categories:

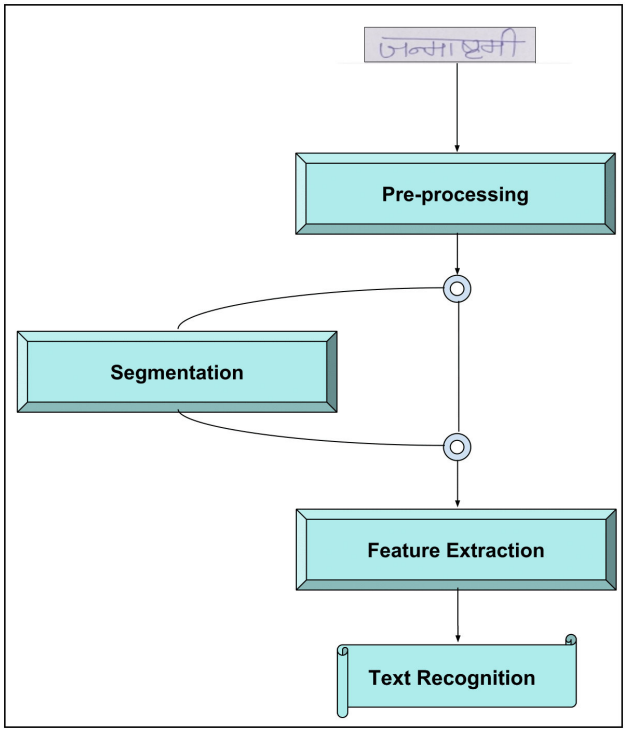

Figure 4.1: A flowchart representing the workflow of a text recognition system. Segmentation may or may not be performed, depending on the solution being used.

- **Conversion to Grayscale or Binary**: This is the process of converting colour or RGB images to either a grayscale [36] image or a binary bit-map [10, 40], generally with white pixels as background and black pixels as text.

- **Noise Removal**: Here we deal with the process of erasing pixels from the input image which hold negligible discriminative power. Noise gets added to the image during the scanning process. Such noise also can interfere with the process of segmentation later on [1]. Gaussian filtering, contour smoothing, contour or boundary extraction, text stroke reconstruction, etc. [59, 60] are some of the methods used for noise removal.

- **Normalization**: refers to the process of reducing the variation across text appearing in various images. The variation may be regarding the written character sizes, different handwriting style across humans, slant, skew, etc. One approach for slant correction is provided by [61], which uses vertical contours to estimate a shear angle by which the image is slanted. The concept of water reservoir is used by [1] to correct for slant. Another normalization procedure that is usually performed is to ensure that all the input tensor values lie between [-1, 1].

4.2.2 Segmentation

Indic scripts have curved cursive characters shapes and are complex inflectional languages, because of which segmentation is often quite challenging. Segmentation of characters/ligature generally requires accurately finding its starting and ending point in the text stroke. Character segmentation is used only by some methods for word recognition like [1]. In general, simple techniques like profile and contour-based methods do not provide accurate results for segmentation with strongly skewed or overlapping handwriting as is the case with handwritten words [62].

In [1], the authors segment Bangla and Devanagari word images into upper, middle and lower zones, using morphology and shape matching to enhance their recognition performance. However another issue with using segmentation for Indic scripts is that it is not language agnostic, we would need to incorporate multiple word to character segmentation techniques for all the different Indic scripts.

4.2.3 Feature Extraction

Feature extraction is a key component of the text recognition pipeline. By extracting meaningful and robust features, we capture the intrinsic characteristics which differentiate one character from the other. However, with the advent of deep learning the use of handcrafted features for text recognition is dwindling. The extracted features are passed on to the recognition stage. The categorization of most common feature extraction techniques is as follows:

- **Structural Features:** Are generally based on geometrical properties of a symbol like loops, directions of strokes, intersections of strokes, and end points. Examples include GSC features [63], connected components [1], etc.

- **Statistical Features:** They are derived from the statistical distribution of pixels in the handwritten image. In zoning, the characters are divided into overlapping and non-overlapping regions and analyzed for pixel density [64]. Another example is histograms of chain code directions that is used by [65]. Other examples of statistical features are moments or projection features [62]. Histogram of Oriented Gradients is also another statistical feature [40].

- **Raw Features:** Use direct pixel values and are generally the most popular with deep learning based solutions [2, 36]. Deep learning based networks like CNN's are known to learn robust features by themselves [66].

4.2.4 Recognition Method

This stage takes in features from the feature extraction stage and either tries to assign a label to it from the given set of classes or predicts a sequence of labels (recurrent neural networks). This stage

requires a training step where annotated input-output mappings are provided. This trained model is then used to predict the word annotations for a new input sample from the test set.

Currently, the recognition methods can be broken down into roughly three categories:

- **Holistic word recognition:** These methods use segmentation-free but lexicon dependent methods which train on recognizing or representing the whole word [48, 49, 65]. They generally use *Hidden Markov Models* (HMM), *Multi Layer Perceptron* (MLP) [67] or CNN's.
- **Character segmentation based:** Here we segment out the characters within the word image and then use an isolated symbol classifier such as SVM or CNN [1, 68, 69]. This approach suffers from the drawback that we have to use script dependent character segmentation algorithms. Also, for better recognition results we need better segmentation algorithms creating a circular dependency here (Sayre's paradox [70]).
- **Seq2seq formulation:** This approach involves using recurrent neural networks which treats word recognition as a prediction problem where both the input and output are treated as a sequence of vectors and we have to maximize the probability of predicting the output label sequence given the input feature sequence [36, 40, 71]. These methods do not require character level segmentation and are not bound to recognizing a limited set of words. Another benefit of a sequence-to-sequence transcription approach lies in the fact that the input sequence can be transcribed directly to the output sequence without the need for a target defined at each time-step through the use of CTC layer.

A novel approach combining robust convolutional features and transcription abilities of RNNs was introduced for English scene text recognition by [27]. Earlier works such as [40] used hand-crafted features which have been shown to be suboptimal to CNN based features. Our solution architecture is inspired by this work with additional enhancements. Since we use Unicode characters as the basic unit of recognition, our solution architecture is entirely language agnostic.

4.3 Components of the Architecture

In this section, we mention all the major building blocks of our Deep CNN-RNN Hybrid network along with describing how the overall network functions.

4.3.1 Spatial transformer Network

The spatial transformer network [28] is an end-to-end trainable layer which can perform an explicit geometric transformation to the input. The transformation parameters are learnt through backpropagation. As seen in Figure 4.2 it has three main components, the localization network, the grid generator and the sampler.

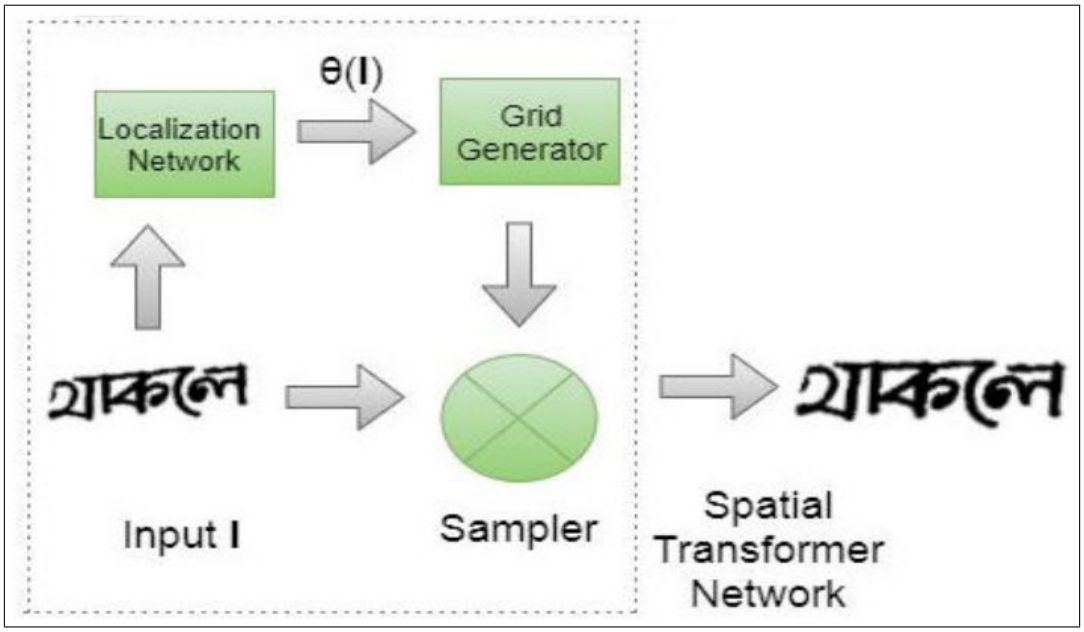

Figure 4.2: The structure of a Spatial Transformer Network (STN). It is made up of three components: localization network, grid generator and sampler.

- **Localization Network**: The localization network takes as input the original grayscale image (or a 3D feature map) $\mathbf{I} \in \mathbb{R}^{W \times H}$, with width W and height H and outputs θ, the transformation parameters to be applied to the input image. The size of θ depends on the transformation being modeled- affine, thin plate spline, etc. For the affine transformation:

$$\theta(\mathbf{I}) = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

- **Grid Generator**: The grid generator generates a grid of coordinates in the input image corresponding to each pixel from the output image $\mathbf{I}' \in \mathbb{R}^{W' \times H'}$. A sampling point (x_i, y_i) on the input image for pixel$(x_{i'}, y_{i'})$ on the output image is computed using the parameters $\theta(\mathbf{I})$ as follows:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \theta(\mathbf{I}) \begin{bmatrix} x_{i'} \\ y_{i'} \\ 1 \end{bmatrix}$$

- **Sampler**: The sampler generates the pixel values for each pixel $(x_{i'}, y_{i'})$ on the output image from the intensity values of pixels (through bilinear interpolation) in the input image which is nearest to (x_i, y_i).

All the above equations are differentiable, which allows the spatial transformer network to be trained using the gradient descent algorithm. In this work, the localization network is modeled as a convolu-

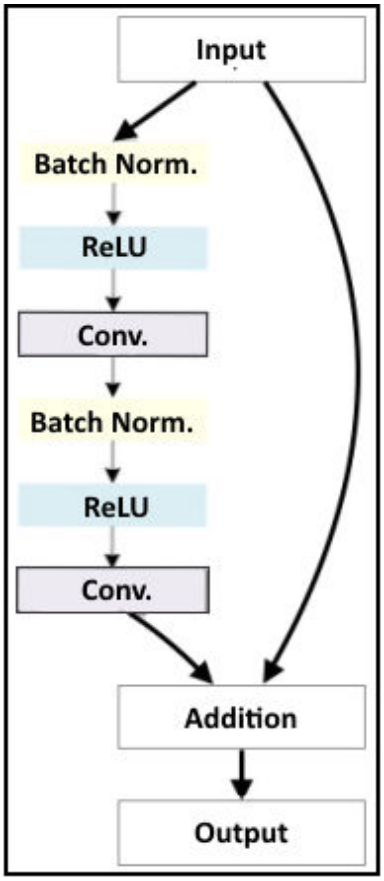

Figure 4.3: The structure of the residual blocks used in the Deep CNN-RNN Hybrid Network.

tional neural network, which includes a final regression layer to predict all the transformation parameters through a fully connected layer of θ dimensions. The network is trained to learn the suitable parameters according to the gradient back-propagated from the CTC objective function directly.

As [72, 73] show, this layer transforms the input feature map such that the geometric transformations are removed from the input and only forwards the relevant part of the input to the subsequent layers. In the case of handwritten images, the role of STN layer is particularly important to handle the variations caused due to the variable hand movements.

4.3.2 Residual Convolutional blocks

Let x denote the input to a convolution block, and $H(x)$ is a complicated function that we want to approximate. A plain convolutional block targets at finding suitable parameters (W_{pb}) of the convolutional layers to approximate $H(x)$ directly, i.e.,

$$H(x) = PB(x, W_{pb})$$

whereas a residual convolutional block, with the special shortcut connection, targets at finding suitable parameters (W_{rb}) of the convolutional layers to approximate the residual function $H(x)x$, i.e.,

$$H(x) = RB(x, W_{rb}) + x$$

Although both blocks are capable of approximating $H(x)$, the ease of optimization is different. It was shown by [20] that adding plain convolutional blocks causes a degradation problem, where the classification accuracy keeps decreasing as we keep adding more plain convolutional blocks. Since our solution uses more than 18 convolutional layers, we employ the residual convolutional block as a basic component to build our deep feature extractor while avoiding issues with training.

The residual block used in our convolutional part of the network includes two convolutional layers, two Batch Normalization [74] layers two ReLU [75] activations and a shortcut connection between the input and the output of the second convolutional layer (see Figure 4.3) [76].

4.3.3 Connectionist temporal classification

A significant challenge of using RNNs for sequence classification tasks is that they require pre-segmented training data and post-processing to transform their output features into a label sequence. Since RNNs are trained to consider prediction at each timestep as an independent event, each input feature needs to be mapped to its corresponding output feature before training the network.

Segmentation of words into their corresponding classes, for complex languages like Indic scripts, is extremely challenging and requires human effort and language knowledge. Hence, we need to use a temporal classifier known as Connectionist Temporal Classification (CTC) [26]. CTC interprets the network outputs as a probability distribution over all the possible label sequences, conditioned on the given input sequence. This probability distribution is used to derive an objective function that maximizes the probabilities of the correct label assignments. It has a differentiable objective function and is trained with standard back-propagation through time (BPTT [77]).

The output activations are normalized such that the resultant on their summing up is one (applying Softmax function). Thus they can be treated as a probability vector of the characters present at that time step. The output layer associates one node for each class label plus another special 'null' character node which indicates a no-character or null label on time steps where no character is determined to be present. Given a training set (S) consisting of paired input and target sequences (x,z), the objective function (O) can be expressed as follows,

$$O = -\sum_{(x,z)\in S} ln\, p(z|x)$$

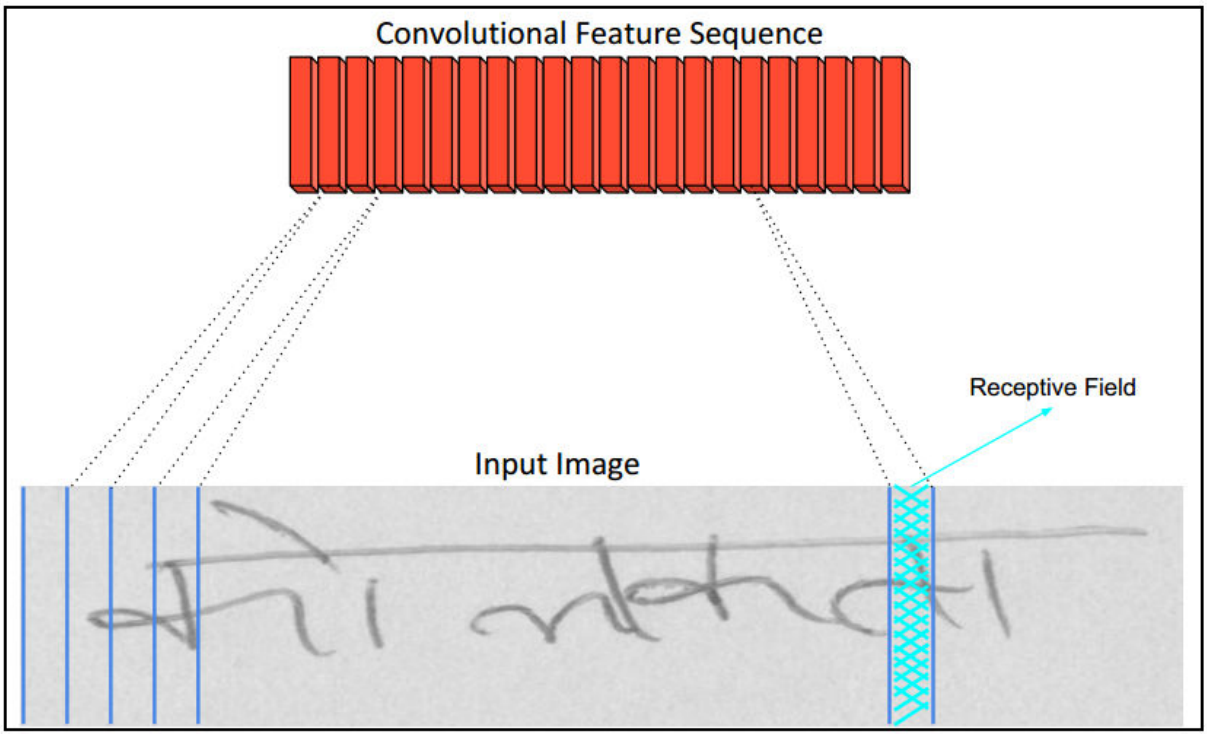

Figure 4.4: Here we see how the convolutional feature sequence relates to the input image. Each column feature is associated with a corresponding column patch on the input image, with the features on the left of the feature sequence corresponding to a receptive field on the left side of the image.

There are two ways in which decoding is done during test time. One is an unconstrained or lexicon free setting where at each time step we choose the character (from the output layer) which gives us the highest probability value. From this prediction, all the null characters are removed along with repeated labels to give us the final predicted label value. In case of constrained or lexicon based decoding, we calculate the output probability O for all the words present in the dictionary and output the word which has the lowest value of the objective function (Here S only contains one image at a time from the test set).

4.3.4 Deep Hybrid CNN-RNN Architecture

Our architecture consists of a Spatial Transformer Network (used for correcting geometric distortion), then a set of convolutional layers, followed by recurrent neural network units whose output is given to a transcription layer which is modeled using connectionist temporal classification. In general, convolutional neural networks have been found to be excellent spatial feature extractors [22, 78, 79] with translation invariant properties while recurrent neural units can take a sequence of feature vectors of variable length and perform sequence to sequence (seq-2-seq) transcription tasks.

In our case, the input sequence of features is constructed from the feature maps of the last convolutional layer by concatenating column features across different channels. These column features act as time steps for the recurrent layers. As the layers of convolution, max-pooling, and activation function operate on local regions, they are translation invariant. Hence, each column of the feature maps corresponds to a rectangle region of the original image and such rectangle regions are in the same order to their corresponding columns on the feature maps from left to right [27]. As shown in Figure 4.4, each rectangular patch on our input image is considered to be the receptive field of its corresponding convolutional feature column vector.

For example, given the feature map of size $512 \times 5 \times 10$, it results in a sequence of 10 feature vectors with dimension $\mathbb{R}^{5\times512}$. Here we choose the column width to be a single pixel. Given a sequence of feature vectors $f_1, f_2, \ldots, f_T$, we forward it to a stacked set of recurrent layers which is our case is a bi-directional LSTM [24] network.

Even though some works in Latin HWR such as [50, 51] use MDLSTM [25] units instead of BLSTM and achieved competitive results, recent works such as [80, 81, 82] have empirically shown that BLSTM units are superior to MDLSTM units for the problem of text recognition. We choose BLSTM over LSTM to be our recurrent unit, since in a text recognition setting, contexts from both directions (left-to-right and right-to-left) are useful and complementary to each other in performing correct transcription.

In our case, the label space consists of the Unicode character set of the given language, plus a blank symbol. Finally, the CTC layer converts the predictions generated by the BLSTM output layer into a maximal probable label sequence for the target language. Figure 4.5 summarizes all the major components of our architecture.

Table 4.1 lists the architecture of our network, along with the details of the convolution and recurrent layers used. For the localization network in our STN, we used three plain convolutional blocks and two linear layers. All the convolutional blocks have filter size, stride and padding of 3x3, 1 and 1 respectively. The number of channels in these layers was 64,64,128. 2x2 max pooling is applied before the first convolutional block and after each convolutional block as well. The first linear layer has 30 units and the second one has six units for learning the parameters of the affine transformation.

To summarize the network, the input image passes through the spatial transformer networks which corrects for the presence of geometric distortion in the input image. Then the residual convolutional layers extract features from the grayscale input image. The BLSTM layers take each feature vector from the feature sequence generated by convolutional layers and process them one by one to make predictions. The sequence-to-sequence transcription is achieved by using a CTC loss layer at the output.

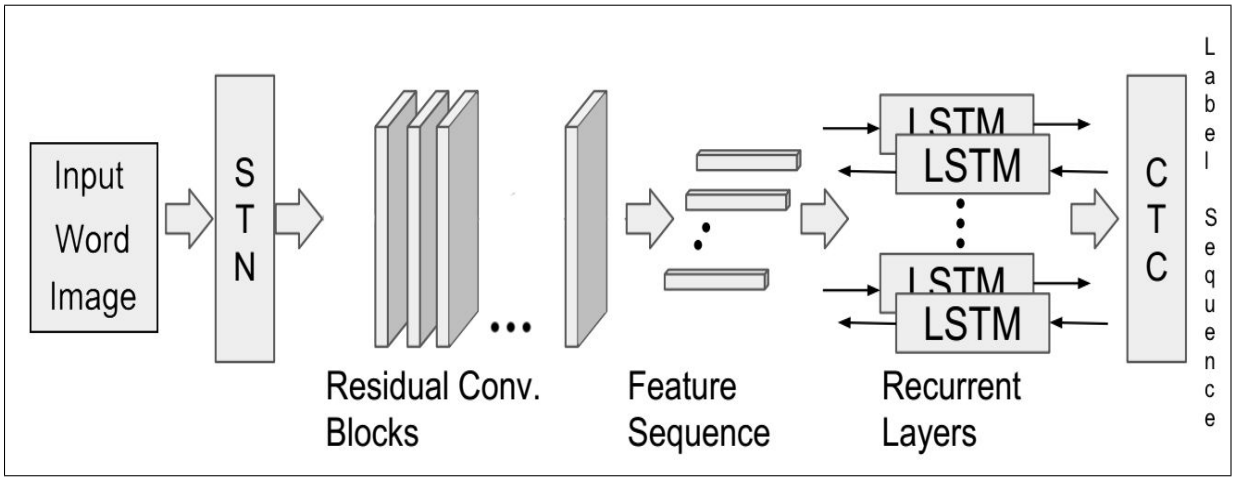

Figure 4.5: Overview of the Deep CNN-RNN Hybrid network architecture used in this work. The various essential components of the architecture are highlighted such as the spatial transformer network, residual convolutional blocks, bi-directional LSTM's and the CTC loss.

Table 4.1: Summary of the network configuration. The width, height and number of channels of each convolution layer are shown in square brackets, with the number of layers that are stacked together. After all but the last block, max pooling is applied. The width and height of the max pooling kernel are shown below each block. The number of units in each BLSTM layer is shown in square brackets, with the number of layers that are stacked together.

Block1 (2x2)	Block2 (2x2)	Block3 (1x2)	Block4 (1x2)	Block5	BLSTM
[3x3,64]x5	[3x3,128]x4	[3x3,256]x4	[3x3,512]x4	[3x3,512]x1	[256]x2

4.4 Training Pipeline

In this section, we describe how we pre-train our networks using synthetic data and do cross-lingual script transfer. We also describe the various data augmentation techniques such as multi-scale and elastic distortion that we used while training our network.

4.4.1 Synthetic Data Generation

The availability of vast amount of data is crucial for the successful training of deep architectures which typically contain millions of parameters. In [29, 83], a framework for rendering synthetic word images from standard fonts is proposed which practically enables building a nearly infinite vocabulary dataset.

In this work, we follow a pipeline similar to [29] for rendering word images for the particular Indic script that we require. We apply various techniques to augment our synthetic data such as applying affine distortion, varying kerning, etc.

4.4.2 Cross-Lingual Script Transfer

Works such as [79, 84] have shown that the filters of lower convolutional layers learn generalized features acting as edge and shape detectors. As we go to the higher layers, the layers learn more specialized features specific to the current task. More recently, [82] reported state of the art HWR recognition results on English handwriting dataset (IAM) by firstly training their network using a dataset that contained Latin languages such as French, English, etc. and Russian, a *Cyrillic* script. After training their network to convergence on this mixed dataset, the authors fine-tuned the model for each language separately.

Inspired by [82], we first pre-train all our models with the IAM datasets and then using synthetic data (based on the final script on which the network was to be trained).

4.4.3 Data Augmentation Schemes

Pre-training a network using data from an unrelated script and synthetic data from the same script gives us a well initialized network to start training on the dataset of focus. We now focus on the various data augmentation schemes which supplement the real data while training and helps with the network learning the desired invariances. While training any deep neural network, it is a common practice to introduce artificial variations in data to make the network robust to intra-class variations and prevent over-fitting. Popular techniques include: random crops, horizontal reflection, random flipping of pixels, and affine transformations such as scaling and translation. In this work, we use four types of augmentation schemes: (i) affine transformation, (ii) elastic distortion, (iii) multi-scale transformation and (iv) test time augmentation (only while testing).

4.4.3.1 Affine Distortion

Under affine transformation, we apply translation, scaling, rotation, and shearing. The second last row of Fig. 4.6 shows different possible variations while performing the above mentioned affine transformations to a word image. Here we restrict rotation to a random amount between $(+/-)5$ degrees, while restricting shearing to $(+/-)0.5$ degrees along the horizontal direction which mimics the skew and cursiveness present in natural handwriting. We perform the translation through padding on all four sides, of up to 20 pixels in any direction, to simulate incorrect segmentation of words. We randomly apply a combination of the above three transformations to an input image.

4.4.3.2 Elastic Distortion

Human handwriting has a high degree of oscillation due to the exertion of non-uniform hand muscle forces while writing. These variations can be captured to a certain extent using elastic distortion which was first proposed in [85] for data augmentation of handwritten digits. We adopt a similar scheme for augmenting our word images. The idea is to generate a random displacement field which dictates the computation of new location to each pixel through interpolation. The displacement field is smoothed using a Gaussian filter of standard deviation σ and scaled using constant factor α. The last row of Fig. 4.6 shows different possible variations created for each word image while performing elastic distortion. We apply this distortion directly to word images as mentioned in [85].

4.4.3.3 Multi-scale Distortion

The idea of multi-scale transformation is to learn to predict characters at multiple scales. The scale of a character is dependent on the context in which it occurs in a word. For example, a word image for "car" resized at a fixed height would have all characters in the same scale while another image corresponding to word "Car" would have a different scale for the first character w.r.t to other characters. The above problem can also be generalized to n-grams occurring at different scales. To learn a scale-invariant classification, we present the images at multiple scales, as shown in the second row of Fig. 4.7, allowing the network to learn these while training. More recently in [86], Wigington et al. also presents a method which addresses this issue by normalizing the scale of all images present in a dataset using profile normalization. Our approach addresses the same issue with the help of data augmentation while training the network.

4.4.3.4 Test-time Augmentation

We also perform test time augmentation similar to [87], where we generate $N = 25$ jittered images, by applying the data augmentation techniques mentioned above, for each input test image. We then average out the final layer output predictions for all the N images before the decoding step.

4.5 Implementation Details

All the input images to the network are converted into grayscale. We use a batch size of 64 for training. In the fourth and fifth max-pooling layers, the pooling windows used are rectangular, instead of the usual square windows as used in standard convolutional networks like [78]. The added advantage of doing this that the feature maps obtained after the convolutional layers are wider and hence create longer feature sequences as input for the recurrent layers. Moreover, rectangular windows yield wider receptive fields (illustrated in Fig. 4.4) which are beneficial for performing transcription among confusing characters since more contextual information is preserved.

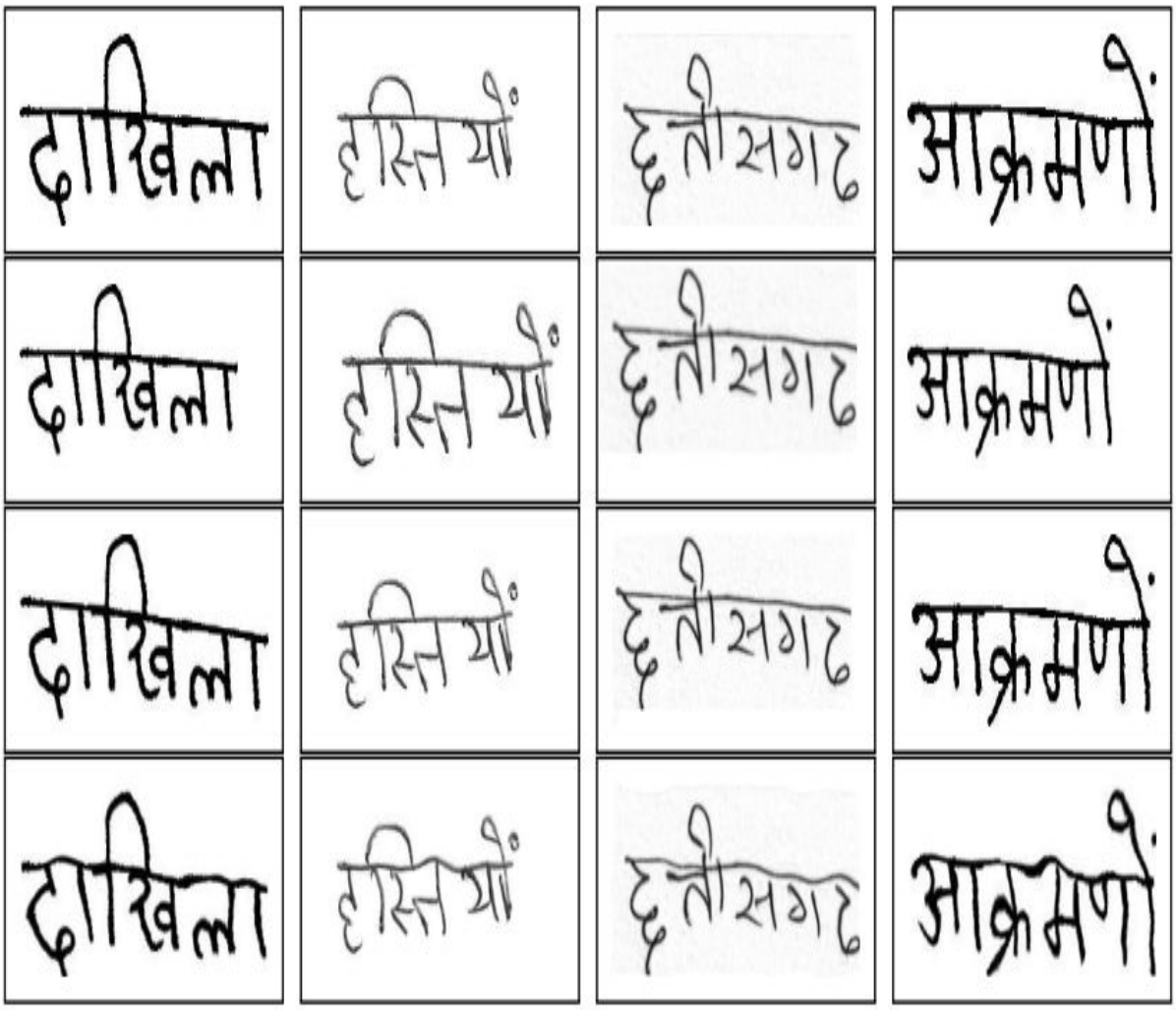

Figure 4.6: Examples of various data augmentation techniques used in this work. The first row shows the original image. The second row shows the image after applying multi-scale augmentation to the corresponding image from the first row. Here the first and the last column images are scaled down while the images in the remaining columns have been scaled up. The third row shows the image after applying affine distortion to the corresponding image from the first row. Here the first and second column images have been rotated, the third column image is translated while the image in the last column has been sheared. The final row shows the image after applying elastic distortion to the corresponding image from the first row.

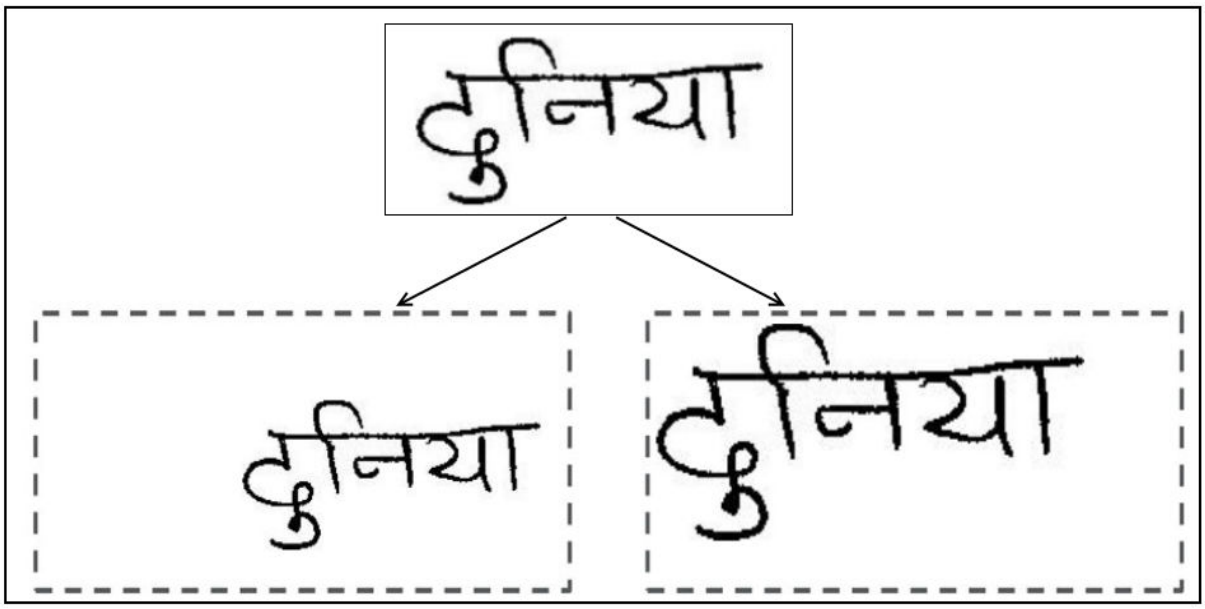

Figure 4.7: A visual illustration of how multi-scaling works. The original image (shown at the top) is scaled up or down and translated at different places in a canvas size of fixed dimensions(which equals the dimensions of the input image to the network.)

To enable faster batch learning, all the inputs are re-scaled to a fixed height (96 pixels) and width (256 pixels), with images which are smaller than this size are padded so as not to cause distortion. However, some images in a dataset inevitably get distorted. Though as per [88] this does not cause much degradation in performance. Training our Deep CNN-RNN Hybrid architecture with the procedure mentioned in the previous section reached convergence on all the datasets in less than or around 20 hours on a single Nvidia-1080Ti GPU. The training took around 11 GB of VRAM and around 40 GB of RAM, running on a machine with dual Intel Xeon E5-2640 v4 processors.

Figure 4.7 gives a visual illustration of how multi scaling works. The original image (shown at the top) is scaled up or down and translated at different places in a bigger canvas size of fixed dimensions(which equals the dimensions of the input image to the network).

The network is trained with *stochastic gradient descent* (SGD) algorithm. Gradients are calculated by the back-propagation algorithm. Precisely, the transcription layers' error differentials are back-propagated with the forward-backward algorithm, as shown in [26]. While in the recurrent layers, the *back-propagation through time*(BPTT) [77] is applied to calculate the error differentials. The hassle of manually setting the *learning-rate* parameter is taken care of by using ADADELTA optimization [89] using zero weight decay, $\rho = 0.9$ and $\epsilon = 10^{-6}$.

4.6 Summary

In this chapter, we did a literature review of the various works in regards to all the components in a HWR system: pre-processing, segmentation, feature extraction and the recognition algorithms themselves. We explained as to why we use a seq2seq based network for HWR. We then did a brief review of all the essential components of the Deep CNN-RNN Hybrid network and explained why they are required. We also explained the overall functioning of our network. Then we moved on to describe how our training pipeline works, with synthetic and cross-script pre-training and data augmentation through various distortions. Finally, we went over some implementation details that might be of interest to research engineers.

Chapter 5

Recognition Results for Indic Scripts

5.1 Introduction

Creating robust text recognition systems for Indic scripts is a challenging task. Other than the common challenges in handwriting recognition due to writer style variation and segmentation, there are a few challenges particular to the recognition of Indic scripts.

One is the presence of modifier and conjunct characters that significantly increases the number of unique character possible in Indic scripts compared to Latin scripts. Secondly, the cursive nature of the shapes leads to confusion between various similarly shaped characters in a language. Finally, there is a lack of publicly available datasets for most of the Indic scripts. This scarcity hinders in the training of modern deep learning based architectures which contain millions of parameters.

In this chapter, we demonstrate the effectiveness of our Deep CNN-RNN Hybrid architecture in recognizing segmented words written in Indic scripts (specifically Bangla, Devanagari and Telugu Indic scripts). Ablation studies are performed to empirically validate our architectural and training choices such as the usage of a deep residual network, synthetic and unrelated script pre-training, spatial transformer network and data augmentation techniques such as elastic distortion, multi scale, etc. We establish the superiority of our solution on a publicly available benchmark dataset for Bangla and Devanagari. We also establish recognition benchmarks for the Devanagari and Telugu datasets that we introduce (IIIT-HW-DEV and IIIT-HW-TELUGU respectively), which we hope will help in establishing these datasets as standard benchmarks.

5.2 Related Works

Most of the earlier methods in the field of Indic script recognition were focused on the domain of printed documents [90, 91]. Various methods involving the k-nearest neighbors classifier, multi-layer perceptrons were tried out for the task. A summary of these works can be found in [92].

There are three popular ways of building handwriting word recognizers for Indic scripts. The first one is to use segmentation-free but lexicon dependent methods which train on recognizing or representing the whole word [48, 49, 65]. These methods fail to recognize any word not present in the lexicon.

Another approach is based on segmenting out the characters within the word image and then use an isolated symbol classifier such as SVM or CNN [68, 69]. In [1], the authors segment Bangla and Devanagari word image into upper, middle and lower zones, using morphology and shape matching. The symbols present in the upper and lower zone are recognized using a SVM while a HMM was used to recognize the characters present in the middle zone. Finally, the results from all the three zones are combined. Figure 5.1 gives a graphical overview of the complete architecture used by [1]. This approach suffers from the drawback that we have to use a script dependent character segmentation algorithms. Also, for better recognition results we need better segmentation algorithms creating a circular dependency here (Sayre's paradox [70]).

The third approach involves using recurrent neural networks which treats word recognition as a seq-2-seq prediction problem where both the input and output are treated as a sequence of vectors and we have to maximize the probability of predicting the output label sequence given the input feature sequence [36, 40]. Earlier works such as [40] used hand-crafted features, but now text recognizers are built using a combination of convolutional neural networks (CNN) and recurrent neural networks (RNN) such as BLSTM's [36]. These methods don't require character level segmentation and are not bound to recognizing a limited set of words.

While our work is similar to that in [36] in terms of the underlying architecture, our network is much deeper, uses residual connections and contains spatial transformation layer to better handle geometric distortions present in handwriting. Also [36] uses *akshara's* as their basic unit of recognition while we choose to use a single Unicode character as our basic unit of recognition.

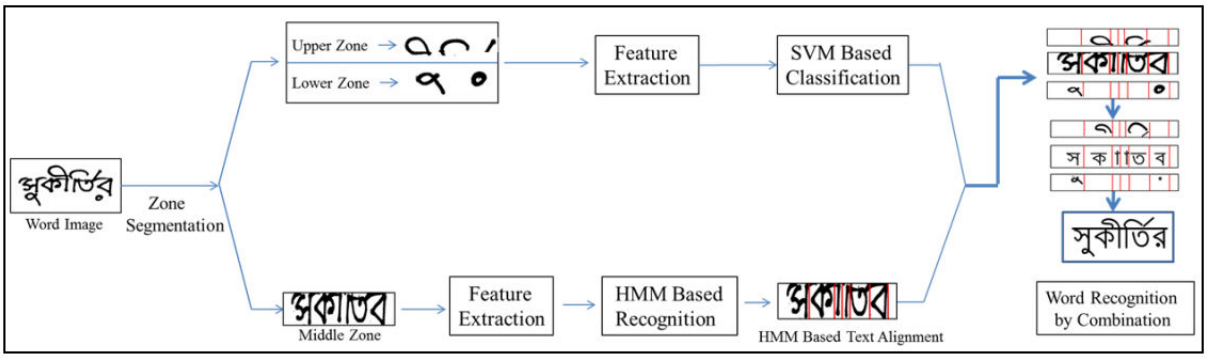

Figure 5.1: Outline of the Handwritten word Recognition framework used by Roy et al. [1]

5.3 Datasets

In this section, we first describe the synthetic dataset and the real handwritten dataset from the Latin script that we use for pre-training our solution architecture. We also talk about the existing publicly available Indic script handwriting recognition datasets (specifically for Bangla, Devanagari and Telugu) that we use for benchmarking purposes.

5.3.1 Synthetic Data and IAM Pre-training

To overcome the lack of annotated data we render synthetic word images. We used 60+ publicly available Unicode fonts for Bangla, Telugu and Devanagari to create our synthetic data. We used the same vocabulary that was as used in the train set of any particular dataset. Here we render the word image in three different ways: without distortion, with a horizontal line at its bottom or with a curved baseline. We applied a varying amount of Gaussian noise and kerning to the rendered images. Also, a small amount of rotation ($(+/-)5$ degrees), shearing between ($(+/-)0.5$ degrees along the horizontal direction) and padding was randomly applied to the generated images. Fig. 5.2 shows a few examples of synthetically generated Telugu word images. Over 1M of such synthetically generated word images were used for pre-training in all the experiments.

In addition to synthetic data, we also use real data from an un-related script which in our case is Latin where there exist large annotated datasets. Similar to works such as [82], we pre-train our model on the IAM train set and present our analysis. After pre-training, our model on the IAM train set the model is pre-trained on the generated synthetic data.

5.3.2 Indic Word Database / RoyDB [1]

It contains samples from over sixty writers and consists of two tracks: Bengali and Devanagari. The Bengali track compromises of 17,901 binarized handwritten word images and the Devanagari track comprises of 16,128 handwritten gray-scale word images. On an average, the label corresponding to a word image in either track consists of 4 characters. We use the word level annotations provided along with RoyDB and follow the standard partition for training, validation and testing. We used the lexicon released as part of the dataset. Figure 5.3 shows a few sample images from all the public benchmark Indic script datasets that are used in this work.

5.3.3 IIIT-HW-Dev [2]

It contains samples from twelve writers and contains 95,381 handwritten word image samples. On an average, each word in the dataset consists of eight basic Unicode characters. Unlike RoyDB this database is labeled using the Unicode. We used the set of all unique words present in this dataset as our lexicon. Details about this dataset were discussed in Chapter 3.

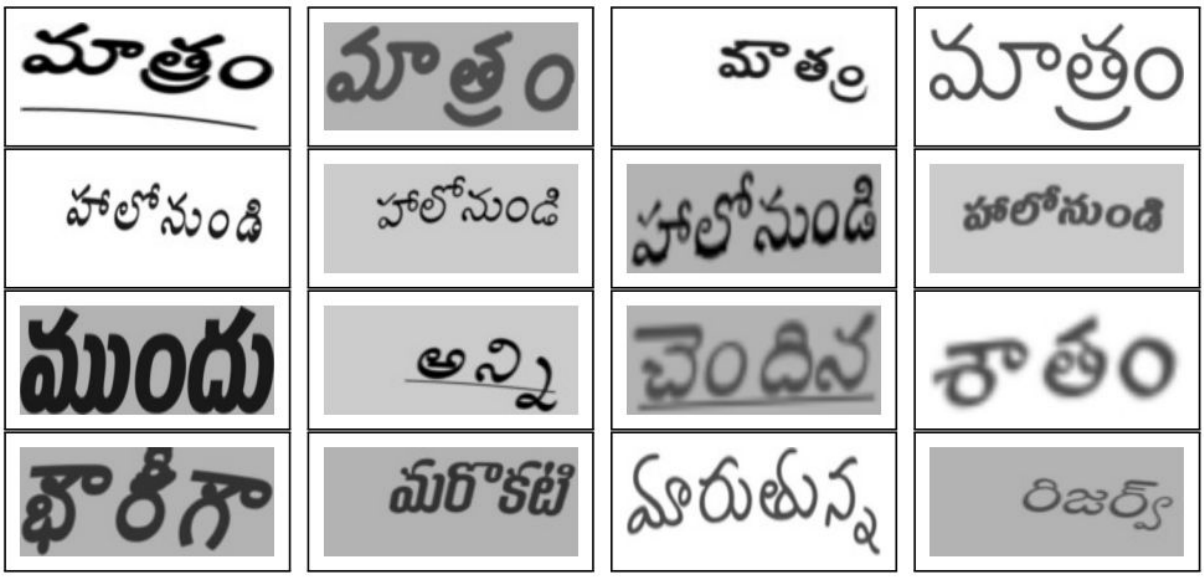

Figure 5.2: Few word images used in the Telugu synthetic dataset. The first two row shows variations for the same word. The last two row shows variations across different words. Similar images were rendered for both Bengali and Devanagari.

5.3.4 IIIT-HW-Telugu [3]

It contains samples from eleven writers and consists of 1,18,515 handwritten word image. On an average, each word in the dataset consists of nine basic Unicode characters. This database is also labeled using Unicode encoding. We used the set of all unique words present in this dataset as our lexicon. Details about this dataset were discussed in Chapter 3.

5.4 Ablation Study on the IIIT-HW-Dev and IIIT-HW-Telugu Datasets

Table 5.1, 5.2 shows the recognition results of various variants of the CNN-RNN hybrid architecture on the test set of IIIT-HW-DEV and IIIT-HW-TELUGU dataset respectively, considering word level segmentation. In this and the next section, we use the standard evaluation measure of Word Error Rate (WER) and Character Error Rate (CER). CER is defined as (where GT: ground truth and PT: predicted text):

$$CER = \frac{\sum_{i \forall samples} EditDistance(GT_i, PT_i)}{\sum_{i \forall samples} \#Chars(GT_i)}$$

and WER for segmented word recognition is defined as the number of predicted words which do not match the ground truth, divided by the total number of words present in the ground truth.

The various models and their training strategies are mentioned below:

Figure 5.3: A few sample images from all the real world handwriting datasets that we use for Indic scripts. The first row contains word images from IIIT-HW-DEV, the second row from IIIT-HW-TELUGU, the third row from ROYDB Bangla track and finally ROYDB Devanagari track.

- CRNN-REAL is the original architecture of [27] and trained only on the IIIT-HW-TELUGU train set with affine distortion.

- CRNN-FULL is the original architecture of [27], first pre-trained on Telugu synthetic data and then fine-tuned on IIIT-HW-TELUGU. Here we only use affine-transformations for augmenting our data.

- S-CRNN-FULL integrates the STN layer and we also apply dropout to the recurrent layers of the above architecture and is trained using the same strategy as above.

- R-CRNN-FULL integrates STN, dropout and residual learning. The network architecture is now the same as the network described in chapter 4, section 4.3.4. It is trained using the same strategy as above.

- R-CRNN-IAM uses the same network architecture as above. It is first pre-trained on the IAM dataset and then pre-trained on the synthetic data. Here we only use affine-transformations for augmenting our data.

- R-CRNN-IAM-DAUG adds the data augmentation strategies of elastic distortion, multi-scale to the above model.

Table 5.1: Ablation study of the CNN-RNN Hybrid architecture on the IIIT-HW-DEV dataset with word segmentation. R-CRNN-IAM-DAUG-TT is referred to as Deep CNN-RNN Hybrid network throughout this thesis.

Method	**WER**	**CER**
CRNN-REAL [27]	27.62	13.45
CRNN-FULL	24.82	11.49
S-CRNN-FULL	22.47	10.34
R-CRNN-FULL	17.36	7.42
R-CRNN-IAM	15.20	6.75
R-CRNN-IAM-DAUG	12.07	5.26
R-CRNN-IAM-DAUG-TT	**11.27**	**4.90**
R-CRNN-IAM-DAUG-TT-LEXICON	**3.40**	**1.52**

- R-CRNN-IAM-DAUG-TT uses test time augmentation along with all of the attributes of the model above. This network and training pipeline is described in Chapter 4 and is referred to as Deep CNN-RNN Hybrid network throughout this thesis.

- R-CRNN-IAM-DAUG-TT-LEXICON uses the same model and training strategy as above but with lexicon based decoding. Here the lexicon consists of the entire vocabulary used in either of the datasets.

Tables 5.1, 5.2 empirically validate the benefit of the various improvements in training and architecture that we have used over the CRNN-REAL network. Both synthetic and unrelated script pre-training are shown are to be an effective way to boost recognition performance. Using dropout in the recurrent layers, a deeper network with residual layers also brings about expected improvement in performance. The effectiveness of the STN layer in correcting distortions present in handwriting can be seen. Both tables also show that using multi-scale transformation (to mimic variation in handwriting scales), elastic distortion (to simulate distortions due to hand movements) and test time augmentation lead to a significant reduction in error rates as compared to just using affine distortions based data augmentation. From the original CRNN-REAL network we progressively reduce the error by a large margin for both datasets.

5.5 Results

We conducted experiments on the publicly available ROYDB [1] dataset, containing both Bengali and Devanagari tracks using the data split mentioned in the dataset. Here we used the same pipeline as R-CRNN-IAM-DAUG-TT, just using the appropriate synthetic and real train data for each track. Table 5.3

Table 5.2: Ablation study of the CNN-RNN Hybrid architecture on the IIIT-HW-TELUGU dataset with word segmentation. R-CRNN-IAM-DAUG-TT is referred to as Deep CNN-RNN Hybrid network throughout this thesis.

Method	**WER**	**CER**
CRNN-REAL [27]	43.25	15.09
CRNN-FULL	39.89	12.67
S-CRNN-FULL	36.78	10.45
R-CRNN-FULL	31.22	8.87
R-CRNN-IAM	28.61	7.28
R-CRNN-IAM-DAUG	24.83	5.05
R-CRNN-IAM-DAUG-TT	**23.98**	**4.58**
R-CRNN-IAM-DAUG-TT-LEXICON	**1.07**	**0.3**

Table 5.3: HWR results on the ROYDB dataset using the Deep CNN-RNN Hybrid network. We used the lexicon released as part of the dataset.

Method	**Lexicon**	**Track**	**WER**	**CER**
Dutta et al. [47]	Free	Bangla	10.71	3.49
Deep CNN-RNN Hybrid Network			**7.04**	**2.55**
Dutta et al. [2]		Devanagari	9.57	3.24
Dutta et al. [47]			12.23	5.17
Deep CNN-RNN Hybrid Network			**8.91**	**3.14**
Dutta et al. [47]	Based	Bangla	4.30	2.05
Adak et al. [36]			14.57	-
Roy et al. [1]			16.61	-
Deep CNN-RNN Hybrid Network			**2.85**	**1.39**
Dutta et al. [2]		Devanagari	4.32	2.07
Dutta et al. [47]			5.13	2.72
Roy et al. [1]			16.61	-
Deep CNN-RNN Hybrid Network			**3.78**	**2.01**

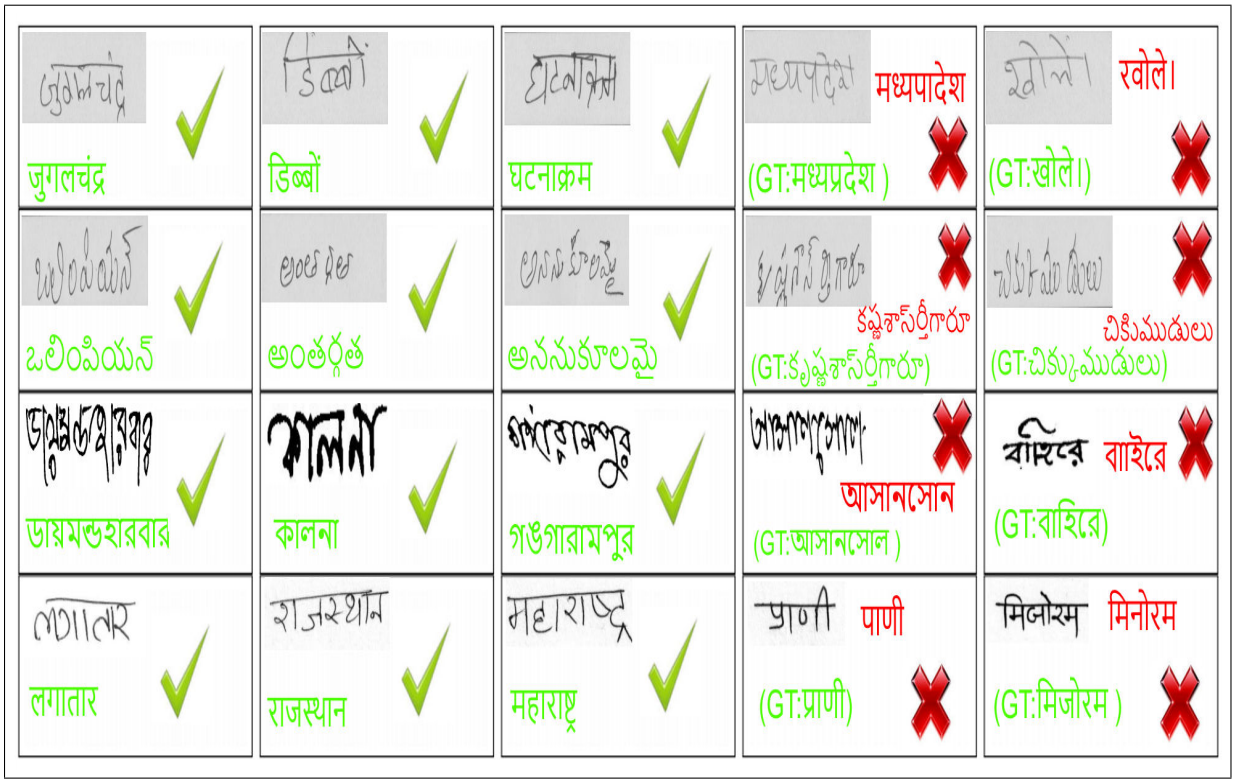

Figure 5.4: Qualitative results of the Deep CNN-RNN Hybrid architecture on the IIIT-HW-DEV (1st row), IIIT-HW-TELUGU (2nd row), ROYDB-BANGLA (3rd row) and ROYDB-DEVANAGARI (last row) datasets. Here GT refers to the ground truth.

report the results for both the tracks of ROYDB. As we can see, our network achieves the state of the art results under all decoding settings.

For all the three datasets, if we use lexicon based decoding, we achieve far better results than the current state of the art in various Latin datasets such as IAM. This observation is most evident in Table 5.2. One reason is that using a lexicon corrects many errors caused due to confusion between modifiers and conjunct characters. To justify this, we take the lexicon of the IAM and IIIT-HW-TELUGU dataset respectively and Fig. 5.5 shows the percentage of valid words that can be converted from one valid word to another, at certain edit distance values, for both datasets. It clearly shows that words in the lexicon of our Telugu dataset are further apart than the words in English. The average edit distance to convert a valid random word from the IAM dataset to another valid word is 7.32 while it is 9.41 for the IIIT-HW-TELUGU dataset. Fig. 5.4 shows the recognized outputs for a few sample word images, from all the four datasets used in the paper, using the R-CRNN-IAM-DAUG-TT model in an unconstrained setting. As we can see, most of the errors were caused by ambiguities in the original word image, due to the subtle differences in the shape of characters.

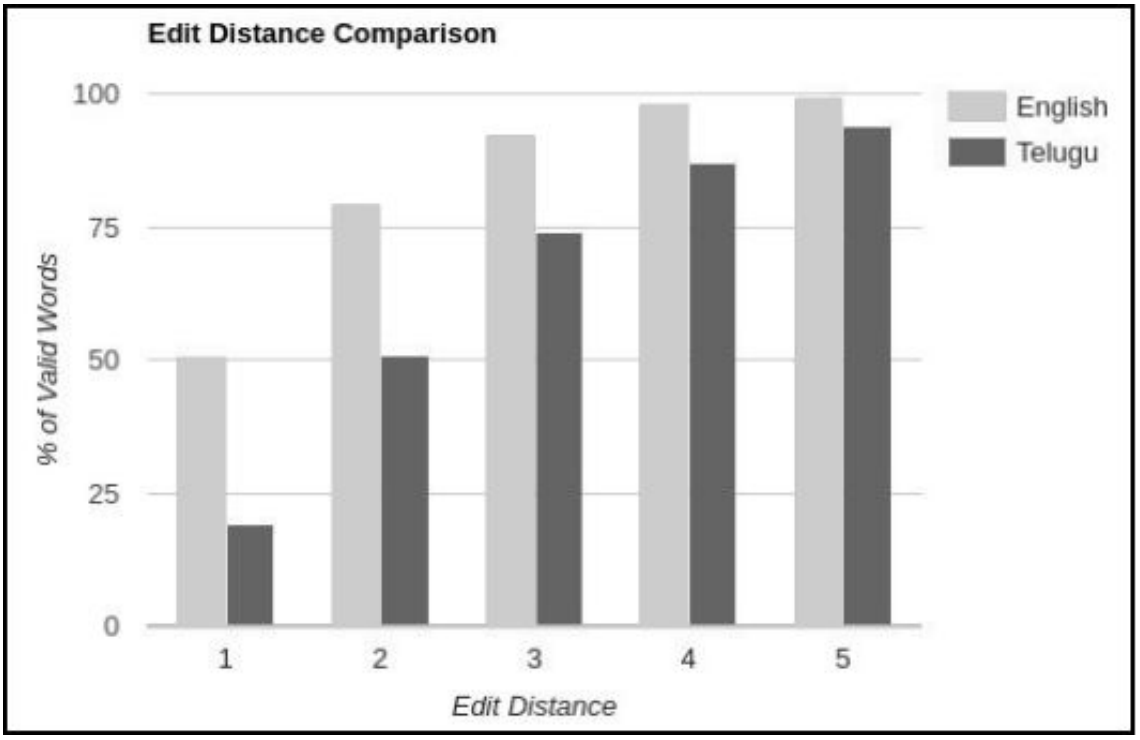

Figure 5.5: Percentage of words that get converted to another valid word in Telugu and English. Here the English and Telugu words sets are from the vocabulary of IAM and IIIT-HW-TELUGU datasets respectively.

If we calculate the out of vocabulary and in vocabulary CER for the ROYDB dataset, we get 18.25 and 2.85 for the Devanagari track. Similarly for the Bangla track we get 13.34 and 2.21. These results show us how deep neural networks tend to overfit with small training data (both tracks had roughly 10k words in their training set). We visualize the convolutional layers of our Deep CNN-RNN Hybrid network using Figure 5.6, where we visualize the activations of a few channels from the second convolutional layer in the first residual block when we give certain images as input to the network. We can see that the channels from this layer act like edge detectors for specific orientations.

5.6 Summary

In this chapter, we first did a brief review of the recent works in the field of Indic scripts HWR. Then we mentioned details about the datasets that we used for pre-training and benchmarking purposes. We performed Ablation studies on the IIIT-HW-DEV and IIIT-HW-TELUGU datasets to empirically validate our architectural and training choices such as the usage of a deep residual network, pre-training, STN and data augmentation techniques such as elastic distortion, multi-scale, etc. We established the superiority of our solution on the publicly available ROYDB dataset.

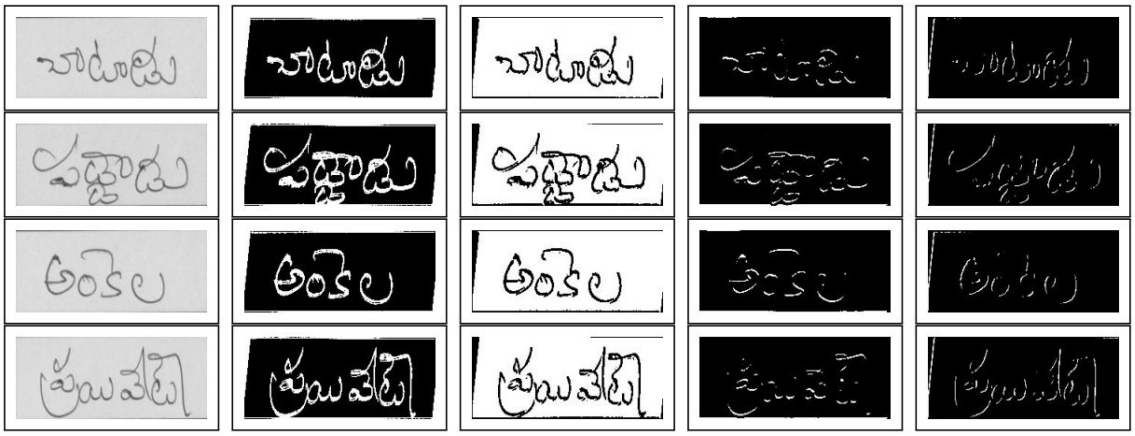

Figure 5.6: Visualizations of the activations in the second convolutional layer in the first residual block of the Deep CNN-RNN Hybrid network. The first column shows the original input image (taken from the IIIT-HW-TELUGU dataset). The second column shows the corresponding activations of a channel which acts as a textual edge detector. The third column shows a channel which activates on the image background. The second last column shows a channel which acts as a horizontal line detector, while the last column shows a channel which acts as a vertical line detector.

Chapter 6

Recognition Results for Latin Scripts

6.1 Introduction

Handwritten word and line recognition in Latin has been extensively researched [24, 50, 57, 93] with a lot of ongoing research [51, 82, 87]. In contrast, such an extensive volume of research is not present for Indic scripts.

In this chapter, we demonstrate the effectiveness of our Deep CNN-RNN Hybrid network in recognizing segmented Latin words in an offline setting. The approach we follow is identical to the approach followed for HWR in Indic scripts. We establish the superiority of our solution on three publicly available benchmark datasets for HWR and thus validate the proposed approach viz a viz other approaches empirically.

6.2 Related Works

Since the start of 2010, most of the research works in Latin recognition have moved away from being HMM based [94, 95] to deep neural networks based works.

The currently popular method of performing handwriting recognition is to model HWR as a seq2seq problem using RNN's and CTC, especially with Latin based works. Earlier works used handcrafted feature extraction methods [96]. With the popularity and robustness of CNNs [76, 78, 79], most works now use convolutional layers as their feature extractors. Most of these CNN-RNN seq2seq architectures are based on the work by [27].

A majority of the current works have variations on the recurrent unit of the network (usually BLSTM) for transcription. Sueiras et al. [97] run a sliding window over the input image, where each patch is given to a convolutional feature extractor and later given to an encoder-decoder BLSTM network with attention for the transcription. Sun et al. [98] use a fully convolutional network as their feature extractor and use multi-directional (MDIR) LSTM's as recurrent units. Voigtlaender et al. [51] use multi-dimensional

LSTM's (MDLSTM) as their recurrent units while also using convolutional layers as feature extractors. Chen et al. [99] use a multi-task network that is able to do both script identification and handwriting recognition simultaneously. They use a variation of the LSTM unit, referred to as SEPMDLSTM. However, recently Puigcerver [80] compared the effectiveness of MDLSTM's and BLSTM's (with CNN's as feature extractors) for HWR and concluded that BLSTM's based networks are better.

On the other hand, some works try novel training techniques to achieve higher performance while using BLSTM's. Wigington et al. [86] use a network similar to [27], with BLSTM's as their recurrent layer. However, they use novel normalization and image distortion strategies and achieved competitive results. Other than doing unconstrained and lexicon based decoding, many works like [51, 80] use language model based decoding to reduce errors, especially in the line level recognition setting.

Stuner et al. [100] used a novel lexicon decoding based architecture involving a cascade of LSTM's. A LSTM's predicted word is accepted or rejected during decoding depending on whether it matches any word in the lexicon. If there's no match, then the process continues over to the next cascade. If all the cascade's predictions are rejected, then we apply Viterbi decoding on the rejected words.

Another set of approach specifically for word-level recognition are inspired by word-spotting works using CNN's. In [87] the CNN architecture evaluates whether a certain n-gram is present in a given portion of the image [87]. Krishnan et al. [66] also used a CNN to learn PHOC like attributes for images and embed the text represented through the PHOC features along with the embedded images into a common subspace. Both these methods are based on the PHOC representation proposed in [101].

A method by Toledo et al. [7] tries to combine both word spotting and recurrent networks for recognizing word level images, by first training a PHOCNET [102] for word attribute embedding and then embedding patches of word images into the attribute space. From the projections in the attribute space, a sequence is created and given to a recurrent network to perform transcription. Figure 6.1 gives an overview of the architecture.

6.3 Datasets

In this section, we first describe the synthetic HW-SYNTH dataset that we created for pre-training our solution architecture and then talk about the existing publicly available Latin handwriting recognition datasets that we have used for benchmarking.

6.3.1 HW-Synth Dataset

Similar to Indic scripts the models used for Latin text recognition are pre-trained using synthetic images. Here we use the HW-SYNTH dataset [103], which compromises of nearly 9M synthetic word

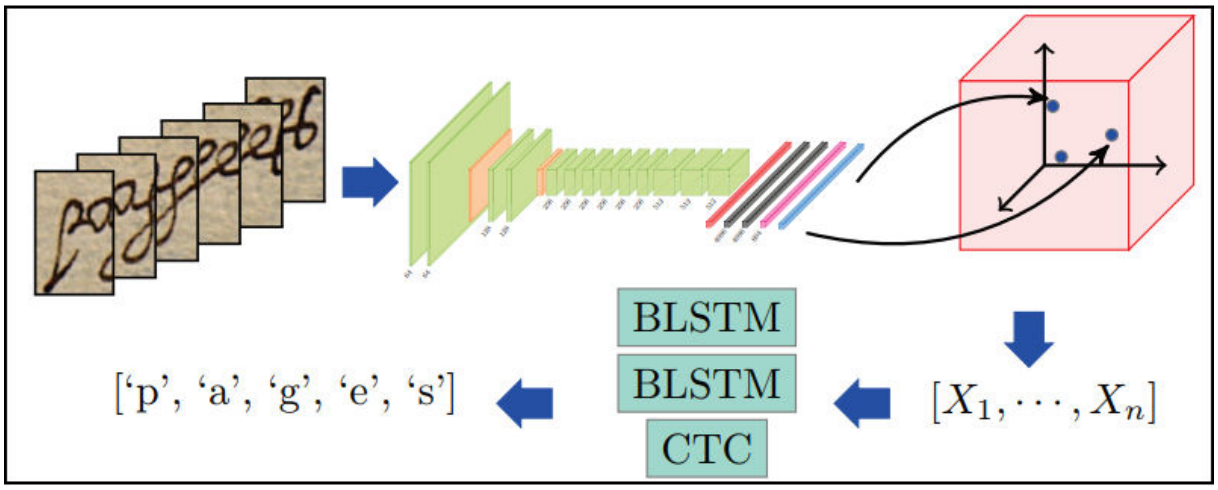

Figure 6.1: System architecture for the work by Toledo et al. [7]. After training a PHOCNET for word attribute embedding, patches of word images are embedded into the attribute space. From these points in the attribute space, we create a sequence that is passed to a recurrent network that performs the transcription.

Figure 6.2: Examples of generated synthetic images in the HW-SYNTH 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.

images rendered out of 750 publicly available handwritten fonts. We use 90K unique words as the vocabulary which is picked from the open source English dictionary Hunspell. For each word in the vocabulary, we randomly sample 100 fonts and render its corresponding image. During this process, we vary the following parameters: (i) kerning level, (ii) stroke width. The foreground and background pixels are sampled from a Gaussian distribution and finally, Gaussian filtering is done to smooth the rendered image.

After rendering the image, we apply affine distortions to simulate real-world variations. A combination of a random amount of rotation, restricted between $(+/-)5$ degrees), shearing restricted between $(+/-)0.5$ degrees along the horizontal direction) and translation through padding on all four sides to simulate incorrect segmentation of words is used.

6.3.2 IAM Dataset

The IAM dataset [19] is currently the most popular benchmarking datasets for English handwriting recognition. It includes contributions from 657 writers, having a total of 13,353 handwritten lines, comprising of 115,320 words. The database is annotated at the sentence, line and word levels. We use the standard partition for training, testing, and validation provided along with the dataset.

6.3.3 RIMES Dataset

The ICDAR 2011 competition version of the RIMES [30] database has contributions from over a thousand writers and has a total of 12,093 lines and 66,982 words. A train, val and test split was released for isolated word recognition as part of the competition which we follow here. For word level recognition, we use a lexicon that was released as part of the ICDAR 2011 competition.

For both IAM and RIMES dataset we do not consider punctuation or capital letters for recognition, similar to [87].

6.3.4 George Washington Dataset

The George Washington (GW) [31] contains 4894 word images written by George Washington and his associates in 1755. We use the first partition of the dataset for all our experiments, same as [7], resulting in 2433, 1293, 1168 word images for training, validation and testing respectively. We keep all the punctuation and capital letters in the database, similar to the setting in [7].

For both the GW and IAM dataset, in one experimental setting, we use a lexicon made up of all the words in the dataset, while in another we use a lexicon made up of only the words in the train set.

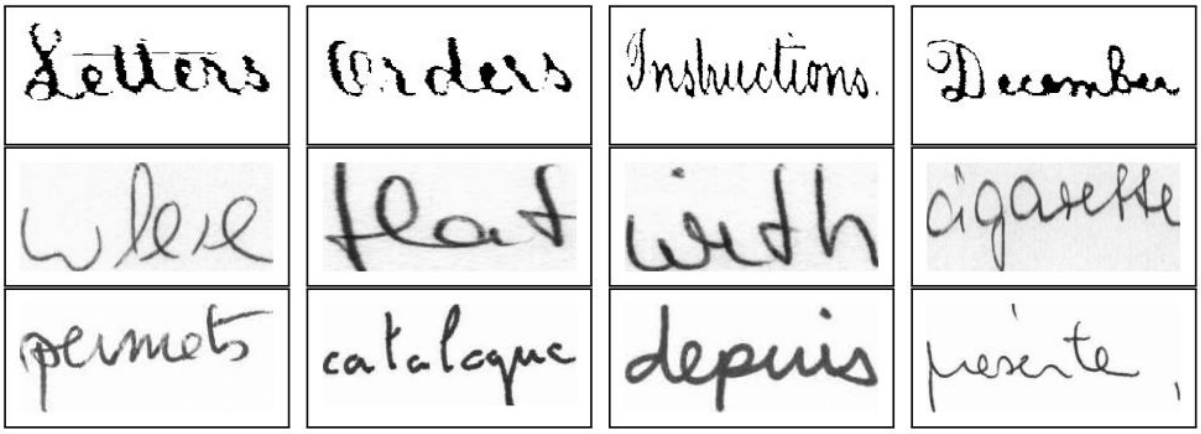

Figure 6.3: Examples of handwritten word images from the three real-world datasets used in this work. The first row shows images from the George Washington dataset, the second row shows images from the IAM dataset while the final row shows images from the RIMES dataset.

6.4 Results

In this section, we showcase the performance of our Deep CNN-RNN Hybrid network (chapter 4, section 4.3.4) in recognizing handwritten Latin words. However, there are two small changes that we made for Latin scripts. First, only real handwriting data from Latin script was used for pre-training. Second, we use the image de-slant and de-sloping technique proposed by [61]. We show results at word level segmentation for the IAM, RIMES and George Washington dataset, with and without lexicon based decoding.

We use the standard evaluation measure of Word Error Rate (WER) and Character Error Rate (CER). CER is defined as (where GT: ground truth and PT: predicted text):

$$CER = \frac{\sum_{i \forall samples} EditDistance(GT_i, PT_i)}{\sum_{i \forall samples} \#Chars(GT_i)}$$

and WER for segmented word recognition is defined as the number of predicted words which do not match the ground truth, divided by the total number of words present in the ground truth.

Table 6.1: Word recognition results on the IAM dataset under different evaluation settings. Here Full-Lexicon refers to the lexicon created from all the distinct words in the database, while Test-Lexicon contains word only from the test set.

Method	**Decoding**	**WER**	**CER**
Krishnan et al. [104]	Unconstrained	16.19	6.34
Wigington et al. [86]		19.07	6.07
Sueiras et al. [97]		23.8	8.8
Mor et al. [105]		20.45	-
Deep CNN-RNN Hybrid Network		**12.61**	**4.88**
Sun et al. [98]	Full-Lexicon	11.51	-
Wigington et al. [86]		5.71	3.03
Stuner et al. [100]		5.93	2.78
Poznanski et al. [87]		6.45	3.44
Such et al. [106]		8.71	4.43
Mhiri et al. [107]		8.83	5.95
Deep CNN-RNN Hybrid Network		**4.80**	**2.52**
Sueiras et al. [97]	Test-Lexicon	12.7	6.2
Wigington et al. [86]		4.97	2.82
Krishnan et al. [66]		6.69	3.72
Krishnan et al. [104]		5.10	2.66
Deep CNN-RNN Hybrid Network		**4.07**	**2.17**

Table 6.2: Word recognition results on the RIMES dataset under different settings. Here Comp. Lexicon refers to the lexicon released as part of the ICDAR 2011 competition.

Method	**Decoding**	**WER**	**CER**
Wigington et al. [86]	Unconstrained	11.29	3.09
Sueiras et al. [97]		15.9	4.8
Mor et al. [105]		11.95	-
Granet et al. [108]		10.96	2.68
Deep CNN-RNN Hybrid Network		**7.04**	**2.32**
Roy et al. [109]	Comp. Lexicon	23.02	-
Wigington et al. [86]		2.85	1.36
Sueiras et al. [97]		6.6	2.6
Stuner et al. [100]		3.48	1.34
Chherawala et al. [110]		3.6	-
Poznanski et al. [87]		3.90	1.90
Mhiri et al. [107]		6.22	3.28
Such et al.[106]		5.68	2.22
Deep CNN-RNN Hybrid Network		**1.86**	**0.65**

Table 6.3: Word recognition results on the GW dataset under different settings. Here Full-Lexicon refers to the lexicon created from the whole vocabulary of the database, while Train-Lexicon only contains words from the train set.

Method	**Decoding**	**WER**	**CER**
Fischer [111]	Unconstrained	-	20
Sfikas et al. [112]		38.38	-
Toledo et al. [7]		-	7.32
Deep CNN-RNN Hybrid Network		**12.98**	**4.29**
Almazán et al. [101]	Full Lexicon	-	17.40
Deep CNN-RNN Hybrid Network		**12.59**	**3.81**
Almazán et al. [101]	Train Lexicon	-	22.15
Deep CNN-RNN Hybrid Network		**29.54**	**12.29**

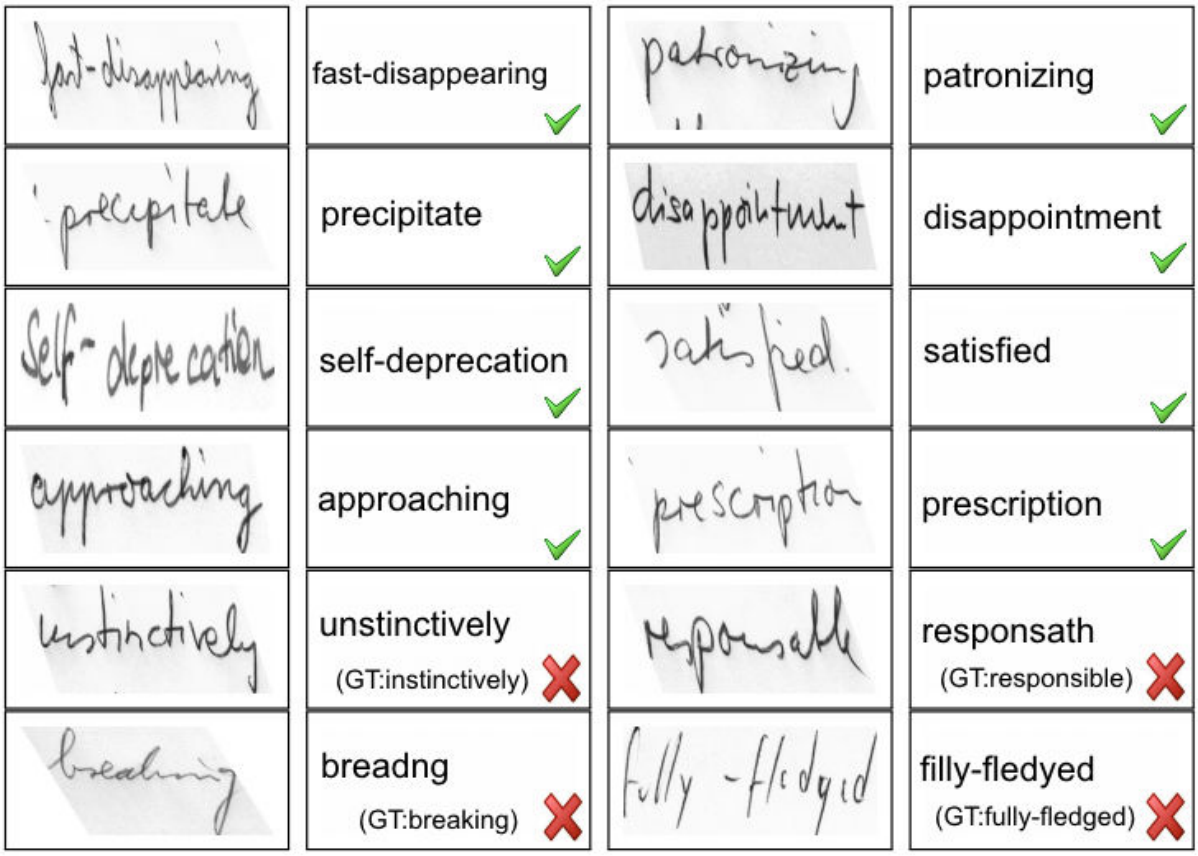

Figure 6.4: Qualitative results of word recognition on the IAM dataset.

Table 6.1, 6.2, 6.3 shows the quantitative word recognition results on the IAM, RIMES and GW dataset respectively, using our Deep CNN-RNN Hybrid network, pre-trained on HW-SYNTH dataset and fine-tuned on the train set of the respective dataset. Here we compare various methods under the presence/absence of lexicon while decoding the output. For all the various decoding settings on the three datasets, we report the state of the art results, even though works like [86, 97, 98, 104], etc. are broadly using a CNN-RNN hybrid network similar to ours. If we calculate the out of vocabulary and in vocabulary CER for the IAM dataset, we get 8.95 and 4.07. These results show us that the network hasn't really overfit on the train set.

Despite the GW dataset being a single writer dataset, we do not obtain nearly as good results for it as compared to the other two datasets. Various factors explain this. First, the small size of the train set in GW and the use of binarized images instead of grayscale images as is the other two datasets. Secondly, none of the old English characters present in GW (old English "G", "E" etc.) are part of our synthetic dataset. We also observe confusion between capital and lower-case letters in the case of lexicon based decoding. The train lexicon in GW has an out of vocabulary (OOV) rate of about 16%, which accounts for our worse performance when using the training lexicon.

Figure 6.5: Qualitative results of word recognition on the RIMES dataset.

Figures 6.4 and 6.5 show the recognized output on a few sample word images from the IAM and RIMES dataset. Here we are decoding using an unconstrained setting. We can observe that most of the errors occur due to the lack of clarity in the original handwritten image regarding the shape of alphabets or due to improper segmentation.

6.5 Summary

We demonstrate how the Deep CNN-RNN Hybrid Network gives state of the art results in offline word level Latin recognition, in both unconstrained and constrained decoding setting. These results should further help in empirically validating the usage of the main components of our training pipeline: 1) Data augmentation, 2) Usage of Synthetic Data for Pre-training, 3) Using a deep residual network with a Spatial Transformer network.

Chapter 7

Conclusion and Future Work

We conclude the thesis by discussing the impact of this work and providing directions for future work that interested readers can pursue.

7.1 Discussion

In this thesis, we analyze the issues and challenges with automatic handwritten word recognition for Indic scripts. We started by giving a brief description of the problem and why we need to solve it in chapter 1. In chapter 2 we gave an overview of the multiple languages and scripts currently in use in India. We talked about the current Unicode encoding standard, why we chose a single Unicode character as our basic unit of recognition. Finally, we talked about the challenges that make HWR for Indic scripts (more characters, lesser real data) more complicated as compared to Latin.

In Chapter 3 we talked about the present lack of HWR datasets for Indic scripts. We also talked about how we collected data for the IIIT-HW-DEV and IIIT-HW-TELUGU datasets by creating forms with QR code embedded boxes. We also talked about how this methodology differs from the conventional way HWR datasets are created. We then went over some statistics regarding both the datasets that were created as part of this thesis.

Chapter 4 started with a detailed literate review of previous solutions proposed for HWR in Indic scripts. We talked about the various pre-processing, segmentation, feature extraction and recognition methods that were used. Then we described the primary building blocks of our solution architecture such as the Spatial Transformer Network, Residual layers, etc. Pre-training with synthetic data and real data from Latin script was discussed next. Finally, the various data augmentation strategies used in this work such as multi-scale and elastic distortion were mentioned.

Recognition results of the network presented in this thesis on various public benchmarks datasets were presented in chapters 5 and 6, for Indic scripts and Latin respectively. We discussed the datasets that were created using the synthetic rendering pipeline. Ablation studies were presented in chapter 5

to showcase the importance of various facets of our network and training pipeline. A brief comparison was made between the different benchmark datasets and the recognition rates of the method presented in this work were shown to be superior to the current state of the art methods on various Latin and Indic script datasets.

7.2 Future Work

Even though Indic script recognition is catching up to Latin script recognition, much progress still needs to be made. A few directions which can be explored for further research in Indic script recognition are listed below:

- **Attention Models**: Have lately received limelight. In a relevant work, [113] introduce an attention-based model that automatically does end to end paragraph based recognition without any segmentation. A similar approach can be used in Indic script HWR, which would be useful for processing unstructured documents.
- **Using an ensemble of CNN-RNN Hybrid and a Word Spotting network**: Here the top k results of the word spotting network like [66] for a test image could be used as a lexicon when the same test image is decoded through the Deep CNN-RNN Hybrid network.
- **Creation of a Historic Focused Collection**: It would pose specific challenges such as historical ligatures, jumbled text, etc. which are not present in modern Indic script HWR datasets. Like the writings of Satyajit Ray, Rabindranath Tagore, etc. Experiments can be made to see how a Deep CNN-RNN Hybrid network performs on such datasets.
- **Using AKSHARA'S as the basic unit of recognition**: With the availability of larger sized corpus of handwritten data, studies should be conducted as to how the performance of a Deep CNN-RNN Hybrid network changes when using AKSHARA'S as the basic unit of recognition instead of Unicode characters.
- **Annotated Datasets**: There are no publicly available annotated dataset present for Indic scripts like Malayalam, Gurmukhi, Odiya. It is essential to have publicly available datasets to stimulate HWR research in these scripts. Bigger sized datasets for scripts where datasets are present would also help increase recognition accuracy.
- **Improvements in Architecture and Training**: Various improvements can be made to the Deep CNN-RNN Hybrid architecture. Using a STN which models a thin plate spline [73] instead of affine distortion, grid-based elastic distortion [86] instead of the one proposed by [85], Image pre-preprocessing using water reservoir technique [1] are a few out of the many improvements that can be made.

Related Publications

- **Kartik Dutta**, Praveen Krishnan, Minesh Mathew and C.V. Jawahar, "*Towards Accurate Handwritten Word Recognition for Hindi and Bangla*", in 6th National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), Mandi, India, 2017. [Oral]

- **Kartik Dutta**, Praveen Krishnan, Minesh Mathew and C.V. Jawahar, " *Unconstrained Handwriting Recognition on Devanagari Script using a new Benchmark Dataset*", in 13th International Workshop on Document Analysis Systems (DAS), Vienna, Austria, 2018. [Oral]

- **Kartik Dutta**, Praveen Krishnan, Minesh Mathew and C.V. Jawahar, "*Improving CNN-RNN Hybrid Networks for Handwriting Recognition*", in 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, USA, 2018. [Nominated for Best Student Paper, Oral]

- **Kartik Dutta**, Praveen Krishnan, Minesh Mathew and C.V. Jawahar, "*Towards Spotting and Recognition of Handwritten Documents in Indic Scripts*", in 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, USA, 2018. [Oral]

Publications Not Part of the Thesis

- Praveen Krishnan, **Kartik Dutta** and C.V. Jawahar, "*Deep feature embedding for accurate recognition and retrieval of handwritten text*", in 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), Shenzen, China, 2016. [Oral]

- Praveen Krishnan, **Kartik Dutta** and C.V. Jawahar, "*Word Spotting and Recognition using Deep Embedding*", in 13th International Workshop on Document Analysis Systems (DAS), Vienna, Austria, 2018. [Best Paper]

- **Kartik Dutta**, Minesh Mathew, Praveen Krishnan and C.V. Jawahar, "*Localizing and Recognizing Text in Lecture Videos*", in 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, USA, 2018.

Bibliography

[1] 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. viii, x, 4, 8, 13, 20, 21, 22, 35, 36, 39, 40, 54

[2] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Offline handwriting recognition on devanagari using a new benchmark dataset," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pp. 25–30, 2018. viii, 21, 36, 40

[3] K. Dutta, P. Krishnan, and C. Mathew, Minesh Jawahar, "Towards spotting and recognition of handwritten words in indic scripts," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018. viii, 13, 37

[4] K. Roy, S. Vajda, U. Pal, and B. B. Chaudhuri, "A system towards indian postal automation," in *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pp. 580–585, IEEE, 2004. ix, 2

[5] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu, "Recognition of numeric postal codes from multi-script postal address blocks," in *International Conference on Pattern Recognition and Machine Intelligence*, pp. 381–386, Springer, 2009. ix, 1, 2

[6] T. V. Lakshmi, P. N. Sastry, and T. Rajinikanth, "A novel 3d approach to recognize telugu palm leaf text," *Engineering Science and Technology, an International Journal*, vol. 20, no. 1, pp. 143–150, 2017. ix, 1, 7

[7] J. I. Toledo, S. Dey, A. Fornés, and J. Lladós, "Handwriting recognition by attribute embedding and recurrent neural networks," in *14th International Conference on Document Analysis and Recognition (ICDAR)*, 2017. xi, 45, 46, 47, 50

[8] T. M. Rath and R. Manmatha, "Word spotting for historical documents," *International Journal of Document Analysis and Recognition (IJDAR)*, 2007. 1

[9] S. N. Srihari and E. J. Kuebert, "Integration of hand-written address interpretation technology into the united states postal service remote computer reader system," in *Proceedings of the 4th International Conference on Document Analysis and Recognition (DAS)*, 1997. 1

[10] S. Thadchanamoorthy, N. Kodikara, H. Premaretne, U. Pal, and F. Kimura, "Tamil handwritten city name database development and recognition for postal automation," in *12th International Conference on Document Analysis and Recognition (ICDAR)*, 2013. 1, 13, 20

[11] R. J. Milewski, V. Govindaraju, and A. Bhardwaj, "Automatic recognition of handwritten medical forms for search engines," *International Journal of Document Analysis and Recognition (IJDAR)*, 2009. 1

[12] C. Adak and B. B. Chaudhuri, "Extraction of doodles and drawings from manuscripts," in *International Conference on Pattern Recognition and Machine Intelligence*, pp. 515–520, Springer, 2013. 1

[13] N. Balakrishnan, R. Reddy, M. Ganapathiraju, and V. Ambati, "Digital library of india: a testbed for indian language research," *IEEE Technical Committee on Digital Libraries Bulletin: Special Issue on Asian Digital Library Research (In Press). Google Scholar*, 2005. 1

[14] Y. Dwivedi, N. Rana, A. Simintiras, and B. Lal, "Digital india programme: a public administration reformation initiative," *Yojana*, vol. 59, pp. 28–34, 2015. 1

[15] "Information access from document images of indian languages," Nov 2015. 1

[16] V. Bansal and R. Sinha, "A complete ocr for printed hindi text in devanagari script," in *6th International Conference on Document Analysis and Recognition (DAS)*, 2001. 2

[17] P. Sankar K, C. Jawahar, and R. Manmatha, "Nearest neighbor based collection ocr," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems (DAS)*, 2010. 2

[18] N. Sankaran and C. Jawahar, "Recognition of printed devanagari text using BLSTM neural network," in *21st International Conference on Pattern Recognition (ICPR)*, 2012. 2

[19] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, 2002. 2, 4, 11, 13, 14, 47

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016. 2, 4, 25

[21] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, 2017. 2

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NIPS)*, 2012. 2, 26

[23] U.-V. Marti and H. Bunke, "Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system," in *Hidden Markov models: applications in computer vision*, 2001. 2, 12

[24] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2009. 2, 12, 27, 44

[25] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in neural information processing systems*, pp. 545–552, 2009. 2, 27

[26] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International conference on Machine learning (ICML)*, 2006. 2, 25, 32

[27] 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," *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2017. 2, 3, 22, 27, 38, 39, 40, 44, 45

[28] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems (NIPS)*, 2015. 3, 4, 22

[29] P. Krishnan and C. Jawahar, "Generating synthetic data for text recognition," *arXiv preprint arXiv:1608.04224*, 2016. 4, 28, 29

[30] E. Grosicki and H. El-Abed, "Icdar 2011-french handwriting recognition competition," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011. 4, 47

[31] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognition Letters*, 2012. 4, 47

[32] "Abstract of speakers' strength of languages and mother tongues - 2001." https://web.archive.org/web/20080404093518/http://www.censusindia.gov.in/Census_Data_2001/Census_Data_Online/Language/Statement1.htm. 6

[33] U. Consortium *et al.*, *The Unicode Standard, Version 2.0*. Addison-Wesley Longman Publishing Co., Inc., 1997. 7

[34] 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*, pp. 1–16, 2018. 8, 13

[35] R. Ishida, "An introduction to indic scripts," in *Proceedings of the 22nd Int. Unicode Conference*, 2002. 8

[36] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive bengali word recognition using cnns with a recurrent model," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. 8, 9, 20, 21, 22, 35, 40

[37] G. B. Kumar, K. N. Murthy, and B. Chaudhuri, "Statistical analysis of telgu text corpora," 2007. 9

[38] M. Mathew, A. K. Singh, and C. Jawahar, "Multilingual ocr for indic scripts," in *12th IAPR Workshop on Document Analysis Systems (DAS)*, 2016. 9

[39] M. Jain, M. Mathew, and C. Jawahar, "Unconstrained scene text and video text recognition for arabic script," in *1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, 2017. 9

[40] U. Garain, L. Mioulet, B. B. Chaudhuri, C. Chatelain, and T. Paquet, "Unconstrained bengali handwriting recognition with recurrent models," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pp. 1056–1060, IEEE, 2015. 9, 20, 21, 22, 35

[41] U. Stiehl, "Sanskrit-kompendium," *Heidelberg: Hüthing*, 2002. 10

[42] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017. 11

[43] E. Sabir, S. Rawls, and P. Natarajan, "Implicit language model in lstm for ocr," *14th International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 12

[44] A. Alaei, U. Pal, and P. Nagabhushan, "Dataset and ground truth for handwritten text in four different scripts," *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 2012. 13

[45] R. Sarkar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "Cmaterdb1: a database of unconstrained handwritten bangla and bangla–english mixed script document image," *International Journal on Document Analysis and Recognition (IJDAR)*, 2012. 13

[46] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Database development and recognition of handwritten devanagari legal amount words," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011. 13

[47] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Towards accurate handwritten word recognition for hindi and bangla," in *Computer Vision, Pattern Recognition, Image Processing, and Graphics: 6th National Conference, NCVPRIPG 2017, Mandi, India, December 16-19, 2017, Revised Selected Papers 6*, pp. 470–480, Springer, 2018. 13, 40

[48] T. K. Bhowmik, S. K. Parui, and U. Roy, "Discriminative hmm training with ga for handwritten word recognition," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008. 12, 22, 35

[49] B. Shaw, U. Bhattacharya, and S. K. Parui, "Combination of features for efficient recognition of offline handwritten devanagari words," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014. 12, 22, 35

[50] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014. 12, 27, 44

[51] P. Voigtlaender, P. Doetsch, and H. Ney, "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. 12, 27, 44, 45

[52] S. Johansson, "The lob corpus of british english texts: presentation and comments," *ALLC journal*, vol. 1, no. 1, pp. 25–36, 1980. 13

[53] D. Goldhahn, T. Eckart, and U. Quasthoff, "Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages," in *LREC*, 2012. 14

[54] S. T. Piantadosi, "Zipfs word frequency law in natural language: A critical review and future directions," *Psychonomic bulletin & review*, vol. 21, no. 5, pp. 1112–1130, 2014. 15

[55] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. 15

[56] P. Soille, *Morphological image analysis: principles and applications*. Springer-Verlag, 1999. 15

[57] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 63–84, 2000. 19, 44

[58] H. El Abed and V. Margner, "Comparison of different preprocessing and feature extraction methods for offline recognition of handwritten arabicwords," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, pp. 974–978, IEEE, 2007. 19

[59] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 2, pp. 216–233, 2001. 20

[60] R. Legault and C. Y. Suen, "Optimal local weighted averaging methods in contour smoothing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 8, pp. 801–817, 1997. 20

[61] A. Vinciarelli and J. Luettin, "A new normalization technique for cursive handwritten words," *Pattern recognition letters*, vol. 22, no. 9, pp. 1043–1050, 2001. 20, 48

[62] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Offline recognition of devanagari script: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 6, pp. 782–796, 2011. 21

[63] B. Shaw, U. Bhattacharya, and S. K. Parui, "Offline handwritten devanagari word recognition: Information fusion at feature and classifier levels," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*, pp. 720–724, IEEE, 2015. 21

[64] S. Kaur, "Recognition of handwritten devanagri script using feature based on zernike moments and zoning and neural network classifier," *A M. Tech. Thesis Report, Panjabi University, Patiala*, 2004. 21

[65] B. Shaw, S. K. Parui, and M. Shridhar, "Offline handwritten devanagari word recognition: A holistic approach based on directional chain code feature and hmm," in *International Conference on Information Technology (ICIT)*, 2008. 21, 22, 35

[66] P. Krishnan, K. Dutta, and C. Jawahar, "Deep feature embedding for accurate recognition and retrieval of handwritten text," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. 21, 45, 49, 54

[67] S. Bhowmik, S. Malakar, R. Sarkar, and M. Nasipuri, "Handwritten bangla word recognition using elliptical features," in *Computational Intelligence and Communication Networks (CICN), 2014 International Conference on*, pp. 257–261, IEEE, 2014. 22

[68] S. Arora, D. Bhattacharjee, M. Nasipuri, L. Malik, M. Kundu, and D. K. Basu, "Performance comparison of svm and ann for handwritten devnagari character recognition," *arXiv preprint arXiv:1006.5902*, 2010. 22, 35

[69] K. Mehrotra, S. Jetley, A. Deshmukh, and S. Belhe, "Unconstrained handwritten devanagari character recognition using convolutional neural networks," in *Proceedings of the 4th International Workshop on Multilingual OCR*, 2013. 22, 35

[70] K. M. Sayre, "Machine recognition of handwritten words: A project report," *Pattern recognition*, vol. 5, no. 3, pp. 213–228, 1973. 22, 35

[71] N. Sankaran and C. Jawahar, "Recognition of printed devanagari text using blstm neural network.," in *ICPR*, vol. 12, pp. 322–325, 2012. 22

[72] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai, "Robust scene text recognition with automatic rectification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 24

[73] W. Liu, C. Chen, K.-Y. K. Wong, Z. Su, and J. Han, "Star-net: A spatial attention residue network for scene text recognition.," in *BMVC*, 2016. 24, 54

[74] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015. 25

[75] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010. 25

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision (ECCV)*, 2016. 25, 44

[77] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, 1990. 25, 32

[78] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision (IJCV)*, 2016. 26, 30, 44

[79] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 26, 29, 44

[80] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, vol. 1, pp. 67–72, IEEE, 2017. 27, 45

[81] M. R. Yousefi, M. R. Soheili, T. M. Breuel, and D. Stricker, "A comparison of 1d and 2d lstm architectures for the recognition of handwritten arabic.," in *Document Recognition and Retrieval XXII*, International Society for Optics and Photonics, 2015. 27

[82] T. Bluche and R. Messina, "Gated convolutional recurrent neural networks for multilingual handwriting recognition," in *Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 27, 29, 36, 44

[83] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *arXiv preprint arXiv:1406.2227*, 2014. 28

[84] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision (ECCV)*, 2014. 29

[85] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis.," in *7th International Conference on Document Analysis and Recognition (ICDAR)*, 2003. 30, 54

[86] 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 *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 30, 45, 49, 50, 51, 54

[87] A. Poznanski and L. Wolf, "CNN-N-Gram for handwriting word recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 30, 44, 45, 47, 49, 50

[88] M. Jain, M. Mathew, and C. V. Jawahar, "Unconstrained ocr for urdu using deep cnn-rnn hybrid networks," in *4th Asian Conference on Pattern Recognition (ACPR 2017), Nanjing, China*, p. 6, 2017. 32

[89] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012. 32

[90] B. Chaudhuri and U. Pal, "A complete printed bangla ocr system," *Pattern recognition*, vol. 31, no. 5, pp. 531–549, 1998. 34

[91] V. Bansal and M. Sinha, "A complete ocr for printed hindi text in devanagari script," in *icdar*, p. 0800, IEEE, 2001. 34

[92] U. Pal and B. Chaudhuri, "Indian script character recognition: a survey," *pattern Recognition*, 2004. 34

[93] H. Bunke, "Recognition of cursive roman handwriting: past, present and future," in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pp. 448–459, IEEE, 2003. 44

[94] A.-L. Bianne-Bernard, F. Menasri, R. A.-H. Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem, "Dynamic and contextual information in hmm modeling for handwritten word recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 10, pp. 2066–2080, 2011. 44

[95] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 4, pp. 767–779, 2011. 44

[96] T. Bluche, H. Ney, and C. Kermorvant, "A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition," in *International Conference on Statistical Language and Speech Processing*, pp. 199–210, Springer, 2014. 44

[97] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," *Neurocomputing*, 2018. 44, 49, 50, 51

[98] Z. Sun, L. Jin, Z. Xie, Z. Feng, and S. Zhang, "Convolutional multi-directional recurrent network for offline handwritten text recognition," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. 44, 49, 51

[99] Z. Chen, Y. Wu, F. Yin, and C.-L. Liu, "Simultaneous script identification and handwriting recognition via multi-task learning of recurrent neural networks," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 45

[100] B. Stuner, C. Chatelain, and T. Paquet, "Handwriting recognition using cohort of lstm and lexicon verification with extremely large lexicon," *CoRR, vol. abs/1612.07528*, 2016. 45, 49, 50

[101] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2014. 45, 50

[102] S. Sudholt and G. A. Fink, "Phocnet: A deep convolutional neural network for word spotting in handwritten documents," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016. 45

[103] P. Krishnan and C. Jawahar, "Matching handwritten document images," in *European Conference on Computer Vision (ECCV)*, 2016. 45

[104] P. Krishnan, K. Dutta, and C. Jawahar, "Word spotting and recognition using deep embedding," in *Proceedings of the 13th International Conference on Document Analysis and Recognition (DAS)*, 2018. 49, 51

[105] N. Mor and L. Wolf, "Confidence prediction for lexicon-free ocr," *arXiv preprint arXiv:1805.11161*, 2018. 49, 50

[106] F. P. Such, D. Peri, F. Brockler, P. Hutkowski, and R. Ptucha, "Fully convolutional networks for handwriting recognition," in *16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018. 49, 50

[107] M. Mhiri, C. Desrosiers, and M. Cheriet, "Convolutional pyramid of bidirectional character sequences for the recognition of handwritten words," *Pattern Recognition Letters*, vol. 111, pp. 87–93, 2018. 49, 50

[108] A. Granet, E. Morin, H. Mouchère, S. Quiniou, and C. Viard-Gaudin, "Transfer learning for a letter-ngrams to word decoder in the context of historical handwriting recognition with scarce resources," in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1474–1484, 2018. 50

[109] P. P. Roy, G. Zhong, and M. Cheriet, "Tandem hidden markov models using deep belief networks for offline handwriting recognition," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 7, pp. 978–988, 2017. 50

[110] Y. Chherawala, P. P. Roy, and M. Cheriet, "Combination of context-dependent bidirectional long short-term memory classifiers for robust offline handwriting recognition," *Pattern Recognition Letters*, vol. 90, pp. 58–64, 2017. 50

[111] A. Fischer, *Handwriting recognition in historical documents*. PhD thesis, University of Bern, 2012. 50

[112] G. Sfikas, G. Retsinas, and B. Gatos, "A phoc decoder for lexicon-free handwritten word recognition," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, vol. 1, pp. 513–518, IEEE, 2017. 50

[113] T. Bluche, J. Louradour, and R. Messina, "Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, vol. 1, pp. 1050–1055, IEEE, 2017. 54