

Enhancing OCR Performance with Low Supervision

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

*Master of Science in **Computer Science** by Research*

by

Deepayan Das
20172143



International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2021

Copyright © Deepayan Das, 2021
All Rights Reserved

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled “Enhancing OCR Performance with Low Supervision” by Deepayan Das, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Dr. C.V Jawahar

To my family

Acknowledgments

I would like to express my deepest gratitude to professor C.V. Jawahar for his constant guidance and support. I am thankful to him for taking a chance on me and allowing to me to be a part of his lab. I am thankful to my wonderful seniors at CVIT- Praveen, Aniket, Surya, Minesh, Aditya, Avijit, Ajoy, Pritish, Rajvi and Saini for for their counsels on research as well as on life in general .

I am also equally grateful to my batch mates Vamsi, Jerin, Anuj, Shyam, Sahil, Nitin and Ashu for their camaraderie.

I would also like to extend my gratitude to the CVIT staff and administrative team- Siva, Rohitha, Ram and Aradhna for helping me with the travel grants, dataset and for making all the other administrative tasks hassle-free.

Finally I would like to thank my family for their constant love, support and encouragement.

Abstract

Over the last decade, a tremendous emphasis has been laid on collection and digitization of a vast number of books leading to the creation of so-called ‘Digital Libraries’. Projects like Google Book and Project Gutenberg have made significant progress in digitizing over millions of books and making it available to the public. Efforts have also been made from the perspective of Indic languages where the task to identify and recognize books from several Indian languages has been undertaken by the National Digital Library of India. Advantages of digital libraries can be manifold. Digitization of ancient manuscripts ensures the preservation of knowledge and promotes research. Books in digital libraries are indexed which facilitates easy search and retrieval. They are easy to store and do not take as much effort in maintenance as their physical counterparts.

One of the most important steps in the digitization effort is the recognition and conversion of physical pages into editable text using an OCR. There are commercial OCRs available like Tesseract and Abby fine reader, however, the ability of an OCR to recognize text without committing too many errors depends very much on the print quality of the pages as well as font style of the type-written text. A pre-trained OCR will invariably make errors across pages whose distribution is different in terms of fonts and print quality from the pages on which it was trained. If the domain gap is too large then the number of error words will be too high which will result in investing significant effort in the correction. Since the books need to be indexed, one cannot afford to have too many word errors in the OCR recognized pages. Thus, a major effort must be spent on correcting the error words, misclassified by the OCR. Manually correcting each isolated error word will incur a huge cost and is infeasible. In this thesis, we look at methods to improve OCR accuracy with minimum human involvement. To this effect, we propose two approaches. In the first approach, we strive to improve the OCR performance via an efficient post-processing technique where we aim to group similar erroneous words and correct them simultaneously. We argue that since a book has a common underlying theme, it will contain many word repetitions. These word co-occurrences can be taken advantage of by grouping similar error words and correcting them in batches. We propose a novel clustering scheme which combines features from both images as well as its text transcription to group error word predictions. The grouped error predictions can then be corrected either automatically or with the help of a human annotator. We show via experimental verification that automatic correction of error word batches might not be the most efficient way to correct the error words and employing a human annotator to verify the error word clusters will be a more systematic way to address the issue.

Next, we look at the problem of adapting an OCR to a new dataset without requiring too many annotated pages. Traditional norm dictates finetuning the existing OCR on a portion of target data. However, even annotating a portion of data to create image-label pairs can be a costly affair. For this, we employ a self-training approach where the OCR is finetuned on its own predictions from the target dataset. To curtail the effects of noise present in the predictions, we include only those samples in the training set on which the model is sufficiently confident. We also show that by employing various regularization strategies we can outperform the traditional finetuning method without the need for any additional labelled data. We further show that by combining self-training with finetuning we can achieve a maximum gain in terms of OCR accuracy across all the datasets. We furnish thorough empirical evidence to support all our claims.

Contents

Chapter	Page
1 Introduction	1
1.1 OCR in Digital Library Setting	1
1.1.1 Challenges in improving OCR accuracy for large document collection	2
1.1.2 Large Collection: A blessing in disguise	3
1.2 Contributions	4
1.3 Thesis Organization	4
2 Background	6
2.1 Introduction	6
2.2 General Pipeline	6
2.3 CNN-RNN architecture for OCR	8
2.3.1 Convolutional Neural Networks	8
2.3.2 Bidirectional LSTMs	9
2.3.3 Connectionist Temporal Classification	10
2.4 OCR Post-processing	11
2.4.1 Post-processing for Indic Languages	12
2.4.2 Post-processing in large document collections	13
2.5 Machine Learning in limited labels setting	13
2.5.1 Unsupervised learning	13
2.5.2 Transfer learning	14
2.5.2.1 Semi-supervised Learning	14
2.5.2.2 Self-training	15
2.5.2.3 Label propagation	15
3 A Low Cost Correction Approach for improving OCR Word Accuracy in a Large Document Collection	16
3.1 Introduction	16
3.1.1 Related Work	18
3.2 Cost Effective Correction	19
3.2.1 Types of Errors	19
3.2.2 Evaluation of Error Correction Effort	21
3.2.3 Overview of the method	22
3.3 Grouping Error Words	23
3.3.1 Features and Clustering	23
3.3.2 Practical Issues in Clustering Algorithms	24

3.4	Dataset and Evaluation Protocols	24
3.4.1	Evaluation Protocol	25
3.5	Experiments, Results and Discussion	26
3.5.1	Experimental Setup	26
3.5.2	Cluster Impurity	26
3.5.3	Results and Discussions	27
3.5.4	Results on Large Dataset	28
3.5.5	Error Analysis	29
3.6	Summary	29
4	Adapting OCR with Limited amount of Labelled Data	30
4.1	Introduction	30
4.2	Empirical Verification Framework	32
4.3	Finetuning for text recognition	33
4.3.1	Results and Discussions	34
4.4	Finetuning across Languages	35
4.5	Dataset	36
4.6	Results and Discussions	37
4.7	Self-training for text recognition	37
4.7.1	Details	38
4.7.2	Regularization	39
4.7.3	Results and Discussions	40
4.7.4	Observations	41
4.8	Hybrid Approach: Self-training + finetuning	42
4.8.1	Results and Discussions	43
4.9	Summary	44
5	Summary and Future Works	45
	Bibliography	48

List of Figures

Figure	Page
2.1 Steps followed in digitization projects. Once the books arrive in scanning centers they are scanned, recognized, proof-read and finally uploaded to servers for easy access by people around the globe.	8
2.2 Features extracted by CNN. Each feature vector focuses on a rectangular region of the image and can be considered descriptor for that region [1].	9
2.3 CNN-RNN architecture used in our work [1]	10
2.4 Presence of skewed lines and non-uniform illumination in a document image due to incorrect scanning. Such degradations are very common and result in maximum word errors.	12
3.1 The proposed pipeline for batch correction process where the error instances are clustered and corrected in one go. For a group of error instances, the correct label is chosen and applied. The correct label can be either chosen by a human annotator (a) or generated automatically (b).	17
3.2 The frequency of unique words in a collection of documents. A subset of words in the collection vocabulary have a very high frequency and accounting for 50% of the words present in the collection. Thus it is safe to assume that if errors occurring in this subset are grouped and corrected in a batch, it can lead to a significant reduction in correction cost (Zipf’s Law [2]). . .	19
3.3 Consistent errors generated by OCR for a given document collection. Each row represents different images for the same word and it’s corresponding OCR prediction in the green text box. We can observe that for similar degradations, the OCR outputs similar error patterns.	20
3.4 Pipeline of proposed batch correction approach. Given word images ($w_1 \dots w_n$) and its corresponding OCR predictions ($l_1 \dots l_n$) we form clusters. Next, the clusters containing error instances are sent for correction. We employ two forms of correction approaches which are shown in (a) when the human editor decides the label for a cluster and in (b) when the cluster label is generated automatically.	21
3.5 Qualitative results of k-means + MST clustering on English dataset. Images, relevant to the cluster are marked correct while the false positives are crossed out.	27
3.6 Result on the unannotated data. We observe that as the number of words in the collection increases the automated batch correction method’s ability to correct the errored predictions improve which is reflected by the increase in OCR accuracy.	28
3.7 Failure cases for clustering on text and image features respectively. Each row in the above figure represents one cluster. The text predictions are depicted in green text box.	29

4.1 (a) Word Images from the collection 1 containing clean-annotated images (b) Word Images from the collection 2 containing partially annotated noisy-degraded images. 31

4.2 Multi-task pipeline for training OCR on two languages simultaneously. Here X and Y refer to dataset belonging to the two languages. The architecture consists of shared convolutional layer weights and separate recurrent layers for each dataset. 36

4.3 A pipeline of our proposed iterative self-training approach. At each iteration model, M_i performs inference and confidence estimation on the unannotated dataset. Top k confident samples is mixed with the labelled data L and the combined samples are noised. The network (M_i) is trained on the combined samples at the end of which we obtain model M_{i+1} . The above procedure is repeated until there is no more improvement in word accuracy on test data. 38

4.4 (a) shows the Validation curves for different finetuning methods (b) Validation vs. Epochs plots for the self-training method with various regularizers on English dataset. 42

4.5 (a) Validation losses vs Epochs for self-train, fine-tune and hybrid approaches on English dataset. (b) Comparison of word recognition rates between finetuning and hybrid approach when different percentages of labelled data is used 43

4.6 Qualitative Result of our Base, self-trained and hybrid model for English (left) and Hindi (right) datasets. Here ST+FT refers to the model trained using the proposed hybrid approach. We observe that there is a systematic decrease in the character errors from the base model to the hybrid model. This shows that the hybrid model is the best performing model as it has the ability to correct the character errors incurred by the self-trained model due to the confirmation bias. Here crosses show the incorrect words while the correct words are denoted by tick marks. 44

List of Tables

Table		Page
3.1	Details of the books used in our work. Here FA refers to the fully annotated books whereas PA refers to the partially annotated books.	24
3.2	Evaluation of costs of each approach proposed in this paper. The numbers reported are relative to Full Typing method. We observe a decrease in cost as we go left to right for each clustering approach for books of a given language. ‘I’ stands for image features, and ‘T’ stands for prediction text.	25
3.3	Relative cost of correction with respect to full typing when no batching is involved.	27
4.1	Details of the data used in our work. The table describes the language of the type of collection from which the line images are used. Annotation refers to whether the data split is annotated or not. Also, the column purpose defines the role of each data split.	33
4.2	Character and Word Recognition results for various finetuning methods. The first row shows the CRR and WRR for the pre-trained model. Subsequent rows contain values for models obtained after finetuning the base model.	34
4.3	Details of dataset used for finetuning across languages	35
4.4	Character and word recognition rates for finetuning and multitask learning on different language pairs	36
4.5	Comparison of word and character recognition rates of CRNN models between baseline and our self training framework.	40
4.6	The breakdown effect of each regularization heuristic on VGG CRNN model	41
4.7	Ablation study of various refinements	41
4.8	Character and word recognition rates at the end of each iterative cycle for both English and Hindi datasets	41
4.9	Character and word recognition rates for different scoring mechanisms on English dataset	41
4.10	Character and word recognition rates for finetuning, self-training and hybrid approach on English and Hindi datasets.	44

Chapter 1

Introduction

Optical character recognition (OCR) is a term used to refer to methods pertaining to the recognition of text from scanned documents. Recognizing text from images remains an active area of research among the computer vision community and it continues to grow. One of the reasons for its continued growth can be attributed to the rising popularity of e-books. Nowadays, people prefer to read books on handheld devices like mobile phone or tablets due to their small size, portability and ability to store a large number of books. Digital copies of books have an added advantage of not being subjected to degradation or wear and tear which their physical counterparts are prone to. Also, they are instrumental in preserving old manuscripts and ancient documents which otherwise are at risk of getting lost forever. The above advantages have created a strong demand to digitize large collections of documents and making them available on the internet for easy access (Digital Libraries). As of a 2015 report Google has digitized over 25 million books [3] whereas project Gutenberg [4] and the Million books project has digitized 60,000 and 1.5 million books respectively.

1.1 OCR in Digital Library Setting

OCR in a digital library setting is quite different from an OCR in a conventional single page setting. Firstly, digital libraries introduce substantially more data in the form of books, manuscripts, magazines etc. Thus it requires considerable manpower to carefully operate such a large project. At every stage of the operation be it a collection of books and sending them to scanning centres or proof-reading for the verification of OCR outputted text, humans supervision is required which can prove to be very cost-intensive. Second, the books collected during large-scale digitization can differ from each other in terms of font style/size as well as age. The OCR is more likely to make errors on books that have been printed decades ago. This happens largely because the ink from such aged documents get eroded at many places. The OCR might not be robust enough to handle such degradation as well as variations which results in the OCR getting confused between similar-looking characters. This deteriorates the quality of OCR outputted text which can, in turn, have a very poor effect on many OCR based applications such as information retrieval, text-to-speech, machine translation etc. Besides, since the number of books in a typical digital

library setting can range anywhere between fifty thousand to several million, significant effort needs to be invested in correcting the wrong predictions. Generally, human annotators are employed who correct each error by typing which can prove to be very expensive and time-consuming. Another method to improve the OCR performance is by finetuning the existing OCR on a portion of annotated target data. Adapting a machine-learning model to new data via finetuning is a widely practised approach. However, it has the potential to incur a substantial cost in terms of manual annotations if the size of data is very large (since finetuning requires labelled data pairs). Thus efficient error correction and OCR adaption techniques are much needed that can reduce the human effort without compromising on the quality of the OCR output text. This becomes the motivation behind our work.

1.1.1 Challenges in improving OCR accuracy for large document collection

There exist mainly two approaches to minimize the number of OCR errors and improve word accuracy.

- *First*, we make use of an efficient error correction module that can automatically detect and correct the error word.
- *Second*, we adapt the existing OCR by finetuning it a portion of the new (target) dataset so that the model's parameters are better adjusted to the new font-size/style and degradations of the printed text.

The above two approaches, although effective measures in improving OCR performance, involve considerable manual effort. To better explain let us look at the *first* approach i.e. minimizing the number of errors via error correction. Most of the existing post-processors either make use of a dictionary lookup or a statistical language model (SLM) to detect and rectify the word errors. Although, they are effective approaches towards error correction yet they do not warrant an absolute reduction in errors which can prove to be detrimental to applications such as search and retrieval.

A naive way to rectify the OCR errors would be to employ a human annotator whose job would be to verify and correct the error words which went unchecked by the post-processor. Chief among such errors are the real word errors (RWES) which are formed when the OCR misclassifies a given word into a valid word. Such errors need to be verified and corrected in isolation (one at a time) by the annotator which tends to be an inefficient and time-consuming approach towards error correction owing to the huge volume of OCR output text in a large document collection.

Similarly, the *second* approach i.e. adapting an existing OCR on a new target dataset via finetuning is (manual) labour intensive as well since it involves annotation of a substantial portion of the new dataset. Besides, traditional finetuning approaches fail to take advantage of massive amounts of unlabelled data which is freely available. We agree that large collection of documents can pose a lot of challenges in terms of the amount of human supervision needed, operation cost and time. However, we hypothesize that the massive amount of data if used intelligently can provide cues to improve the performance of

an existing OCR model without needing too much manual intervention. In the following section, we expand more on this hypothesis.

1.1.2 Large Collection: A blessing in disguise

In large digitization efforts, books and manuscripts are sent as a whole through the digitization pipeline. This is in contrast to OCRs in single page setting where text recognition is performed on isolated pages. As a result of which, massive amounts of data gets introduced into the system. We propose that if the large scale data is carefully regularized then it can provide supervisory signals which can help us improve the performance of our OCR at no expense of human effort. Usually, books or manuscripts consist of an underlying theme which leads to a repetition of domain-specific words such as proper nouns, abbreviations etc. Also, if such words are exposed to similar degradation then the OCR will make consistent errors across all such repetitions. For example, if a book consists of the word 'Arjun' and the OCR misrecognizes it to be 'Arjum' then all instances 'Arjun' will appear as 'Arjum' in the OCR outputted text. Therefore, if all such similar error words can be grouped and corrected simultaneously, it can drastically help reduce the human effort. This assumption becomes the basis for Chapter 3 where we group the error words based on image and text features. We then correct each group by propagating the correct label instantly to all the error words present in the group. The selection of the correct label can either be done automatically or with the help of a human. In our work, we show the advantages and disadvantages of both the methods. Ideally, automatic error correction should help us achieve minimum cost in terms of human effort. However, in a real-world scenario, this rarely happens. This is chiefly due to the impurities that might be present in the error word clusters which results in wrong label getting passed onto the rest of the cluster elements. In Chapter 3 we delve into much more detail on the issues of cluster impurities and its effect on annotation cost. However, from our experiments we observed that as the size of data increased, our automatic error correction algorithm became more and more confident in picking the correct label, thus reducing the overall correction cost. Thus the size of the data played a crucial role in alleviating the OCR accuracy by correcting wrong predictions with minimal human effort.

In Chapter 4, we look at another way the large data size helped us to enhance the OCR performance. We did this by adapting the existing OCR to the new unlabelled data by iteratively finetuning it on its predictions (self-training). Traditional finetuning methods require a substantial portion of target data to be annotated which can incur a huge cost. Additionally, such methods do not take advantage of the unlabelled data which is something that we have very easy access to and that too in large quantities. Thus, we look at semi-supervised learning algorithms which can leverage both labelled and unlabelled data to improve the performance of our OCR. Although training the OCR model on its prediction might sound counter-intuitive at first, we provide ample empirical evidence that self-training can help us gain a significant improvement in OCR accuracy.

In this thesis, our goal is to improve the performance of the OCR with a focus on reducing the manual effort. For this purpose, we take inspiration from the machine learning literature with an emphasis on learning under a limited label setting. We propose various self-supervised and semi-supervised approaches to achieve the above-stated goal. In the following sections, we discuss in detail the contributions of this thesis as well as describe the layout in which this thesis is arranged.

1.2 Contributions

In this thesis we explore self-supervised and semi-supervised methods improve the performance of an OCR on large document collections. The major contributions of this thesis are as follows:

- **Batch error correction.** To make the process of human aided error correction time and cost-efficient we propose a method of correcting the error word in batches rather than in isolation. In Chapter 3 we provide details of our batch correction scheme by discussing the algorithms and the features that we use to group and correct the error words. We introduce two variants of the batch correction method 1) Automated 2) Human aided and provide a quantitative comparison between the two. We provide empirical verification for all our experiments and show that the proposed human aided batch correction method achieves 70% reduction in time/cost. We also show that the automated variant of batch correction improves with the size of document collection and we further reason that it can become a viable option for error correction if we have a sufficient number of books in our collection. We present our experimental verification on multiple languages, multiple OCRs and a collection of 100 books.
- **OCR with limited labelled data.** In Chapter 4 we show that by iteratively training a pre-trained OCR on its confident predictions (self-training), we can adapt the OCR to the target dataset at zero additional expense in terms of annotations and achieve results comparable to various traditional finetuning techniques. To achieve this we suggest various regularization techniques to prevent the model from getting biased towards the erroneous predictions which get introduced into the training set. We also show that by combining the self-training and finetuning strategy we can outperform models that have been trained exclusively using the finetuning method. We empirically support our claim on English and Hindi datasets and show that our proposed approach is language independent.

1.3 Thesis Organization

This thesis is organized as follows: Chapter 2 provides an overview of techniques involved in document image to text transcription. We discuss the various components of our OCR model. We lay special emphasis on the various post-processing approaches involved in large document collection as well as in the Indic language setting which are relevant to the thesis. We also provide details on the types of

machine learning approaches under limited labels setting such as transfer learning and semi-supervised learning.

In Chapter 3 we present the problem of error-correction of OCR misclassified words and discuss the relevant works in the literature to address the issue. We list the disadvantages of correcting error words in isolation and propose batch correction of similar error words as a possible remedy. We discuss the various clustering algorithms to group the error words as well as the different features on which we perform clustering. Besides, we also present automated as well as a human aided approach towards the correction of error words. We discuss the advantages and disadvantages of both the approaches and provide empirical evidence for all our claims.

Chapter 4 concerns with finetuning an existing OCR on a new dataset in a limited labels setting. We compare various finetuning techniques with our proposed method on English and Hindi datasets. We also discuss the drawbacks of training an OCR on its predictions and present the various regularization techniques to alleviate limitations posed by noisy predictions in the training data. We also provide an ablation study showing the effect of each regularizer in our self-training framework. Finally, we also show that by combining our self-training method with the traditional finetuning approach we can achieve the best result on both the datasets.

Chapter 5 concludes the discussions of this thesis by consolidating all the contributions made by our work. We also leave pointers on the possible extensions of this field for any interested researchers of the community to pursue.

Chapter 2

Background

2.1 Introduction

In this section, we give a general overview of the various components involved in a large scale digitization process. We follow it by giving an overview of various components present in our OCR model which forms the backbone of all the experiments done throughout this work. Next, we briefly discuss the error correction methods both in general as well as those employed by the large scale projects like Google Books and Project Gutenberg and also mention why such error correction methods are often not scalable. Next, we talk about finetuning an OCR under limited labels setting which falls in the purview of semi-supervised domain adaptation. We discuss some of the relevant works in semi-supervised learning and domain adaptation and emphasize the self-training approach to semi-supervised learning which is the main methodology behind the second half of our thesis.

2.2 General Pipeline

Digitization efforts like the Google books project comprise of individual components that work seamlessly to facilitate the access of millions of books all around the world. Some of the major modules are listed below:

- **Scanning.** Digitization starts with the arrival of books in scanning centres where each book is scanned using a flatbed scanner.
- **Pre-processing.** Post scanning, the page images go through a pre-processing step which involves noise removal and data normalization using the various state of the art image processing techniques. As the name suggests, noise removal is the process of eliminating unwanted artefacts from scanned images. In addition to noise removal, contrast equalization, image sharpening, binarization and deblurring techniques are also performed to bring out the text more vividly against the white background [5]. Data normalization is the process of wrangling the text by stretching and resizing to make it more consistent throughout the pages [6].

- **Segmentation.** Segmentation is dividing the document into smaller components such as lines, words or characters. It is one of the fundamental stages of text recognition as it helps to extract the typewritten words, letters which can then be recognized by the OCR. Character level segmentation is difficult since it requires the knowledge of the starting point and ending point for each character. Most of the recent work in OCR focus on either word level or line level segmentation.
- **Feature Extraction** Text recognition comprises of two main steps. The first step is the extraction of features from the segmented line, word or characters while in the second stage the extracted features are assigned a specific label. Usually feature extractors focus on features that can distinguish between different characters. They rely mainly on the geometrical properties of characters such as shape, width, height etc. [7]. Features can also be derived from a statistical distribution such as moments or projection features. Using pixel values directly as features is a popular choice in the deep learning era. Algorithms like CNNs and RNNs are known to learn features directly from the raw features [8].
- **Recognition** During recognition, we perform classification of extracted features which can either be done using pattern matching algorithms or artificial neural networks. The latter method has become more popular since the onset of deep learning era. Deep learning (DL) algorithms such as Convolutional Neural Networks (CNN) [9] and Long Short-Term Memory Networks [10] (LSTM) can learn the features automatically from pre-defined images and their corresponding labels over the course of many iterations. The trained deep learning model is then used to decipher the labels from unseen images. The output is in machine-readable ASCII format that can be understood by the computer. Recognition can be divided mainly into two categories segmentation and segmentation free approach. Segmentation based approaches as the name suggests divides each word or line image further into its constituent characters followed by feeding them to an isolated symbol classifier such as SVM or a multi-layer perceptron (MLP) [11]. Segmentation free approaches treat the problem of recognition as a sequence to sequence formulation, where a DL based model tries to map the input sequence of features to the most likely output sequence. One of the major advantages of such an approach is that the input sequence can directly be transcribed to an output sequence with the help of the CTC [12] based transcription layer and there is no need of defining target at each time step.
- **Proof-reading.** Post-processing is the final stage of the OCR pipeline which deals with the detection and correction of words which have been misrecognized by the OCR. Error detection typically involves looking up a pre-defined vocabulary to tell whether a given word is valid or not. Error correction the other hand can either be done manually by human proof-readers or automatically by using context-sensitive error correction algorithms that take into account the grammatical and syntactical properties of the specific language to correct the error word. Each corrected text document then undergoes verification by a human annotator/proof-reader.

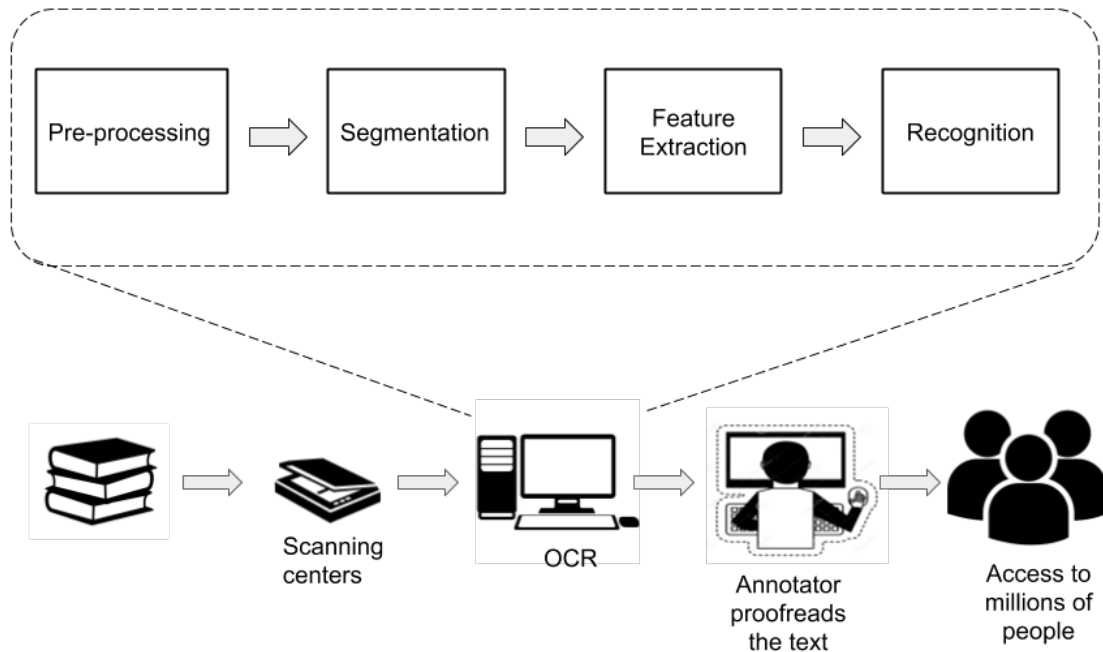


Figure 2.1 Steps followed in digitization projects. Once the books arrive in scanning centers they are scanned, recognized, proof-read and finally uploaded to servers for easy access by people around the globe.

Once the books have been verified and corrected, they are finally hosted over internet servers for access by millions of people.

2.3 CNN-RNN architecture for OCR

In this section, we mention the different components our CNN-RNN architecture which we use as our primary OCR throughout this work. Our network consists of 7 convolutional layers followed by a couple of Bidirectional-LSTMs (BLSTM)s for label prediction and on top, we have a CTC based transcription layer which aligns the target with the output to calculate the loss. The architecture is heavily borrowed from [1] where it was used to achieve substantial gains on scene text data. We depict the architecture in Figure 2.3.

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been widely used in computer vision tasks such as segmentation, object detection and recognition. First proposed in [9] to recognize digits, CNNs have be-

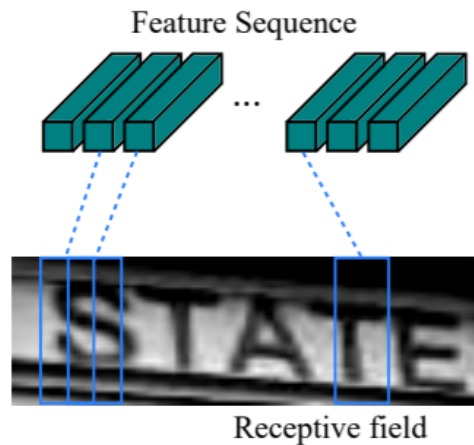


Figure 2.2 Features extracted by CNN. Each feature vector focuses on a rectangular region of the image and can be considered descriptor for that region [1].

come the industry standard in areas related to deep learning. The widespread popularity of CNNs is due to their ability to learn robust, translation-invariant features with very limited parameters as compared to fully connected (FC) neural networks.

The CNNs produce sequence of feature vectors which act as an input to the recurrent layers. Each feature vector from the sequence represents a rectangular region on the original image as shown in Figure 2.2. Thus, the feature vector can be considered to a descriptor for that region. A sequence of all such feature vectors gives a holistic representation for the entire image. In addition to the convolutional layers, our network also consists of 4 max-pooling layers, 3 batch normalization layers [13] and uses ReLU [14] as intermediate activations.

2.3.2 Bidirectional LSTMs

Recurrent Neural Networks RNNs just like CNNs can be considered to be another variant of the traditional feedforward networks with the exception that it keeps track of its hidden state activations and uses it to processes the next input in a sequence. This makes it very useful in cases when the input data is sequential i.e the information carried by the input signal is ordered or time-dependent. Besides, RNNs also can handle variable-length data. The above two properties make RNNs a highly desirable choice for OCR. However, traditional RNNs suffer from the problem of vanishing gradients [15] where the gradients diminish substantially while propagating through the length of the sequence. To combat vanishing gradients LSTMs were introduced in [10], which have internal gating mechanisms that decide the relevance of information from the past hidden states. Thus, only the relevant information is allowed to take part in the decision-making process while the irrelevant information is discarded. The selective nature enables the LSTM to remember long term dependencies. LSTMs were first used for recognition of hand-written documents in [16]. Since, contexts from both the directions are useful in image sequences,

Sankaran et al. [17] showed that a Bi-directional LSTM is more effective for recognizing Devanagari text images. Further, in [18], the authors made significant improvements in recognizing Oriya text by stacking three layers of Bi-directional LSTMs (BLSTMs). The deep structure allows the network to learn a higher level of abstractions than a shallow one. Our network consists of two stacked BLSTMs on top of the CNNs. The recurrent layers take feature sequence from the CNNs as input and predict a label distribution overall feature vectors.

2.3.3 Connectionist Temporal Classification

In sequence-based tasks such as speech and text recognition, it is very important for the input and output sequence to aligned. Proper alignment leads to efficient loss computation between the network predictions and expected output. In the case of segmentation based OCR approaches, there exists a direct one-to-one mapping between the segmented images of characters and the output labels. However, as discussed in the prior sections, such bijective mappings are difficult to achieve. Thus, CTC [12] based transcription layers have become the de-facto choice for OCRs and speech recognition module since it allows loss computation without explicit mapping between the input and output. The CTC layer takes the output from the LSTMs and computes a score with all possible alignments of the target label. The OCR is then trained to predict a sequence which maximizes the sum of all such scores.

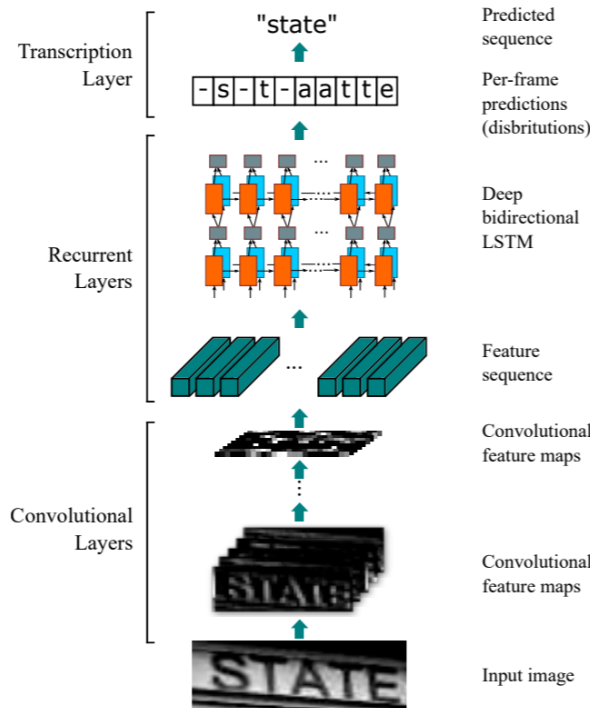


Figure 2.3 CNN-RNN architecture used in our work [1]

2.4 OCR Post-processing

Machine Learning (ML) are not perfect. It is rare for a ML based model to achieve a hundred per cent accuracy on data it was not trained on and OCRs are no exception. There can be numerous reasons for an OCR to misclassify a word. For instance, the page image might contain artefacts which makes it difficult for the OCR to discern the actual word. Some of the major cause of word errors are as follows:

- **Background Noise.** Improper scanning techniques introduces background noises. When books are scanned at such a large scale especially with human supervision, it natural for a few pages to get improperly scanned. Poor scanning also results in the introduction of various artefacts like skew and distortions which poses a challenge to the OCR. Figure 2.4 gives an instance of a document image which is not properly scanned.
- **Document degradation.** A large part of the digitization effort is comprised of books that were printed decades ago. As such, printed characters often suffer from degradations like *cuts* and *merges*. *Cuts* refer to instances when the ink from the characters get eroded which results in a single character being split into two or more segments. On the contrary *merges* refers to a situation where two or more characters seem to merge into one due to ink leakage. Such degradations often result in the OCR to confuse between two similar-looking characters and thus results in the wrong prediction.
- **Font style/size.** It is also common for the documents to have font style/size different from the images on which the model was trained. Thus the model fails to generalize to the newer font type which leads to a loss in the accuracy.
- **Morphologically rich languages.** This mainly concerns non-Latin languages like Hindi, Tamil etc. One of the main challenges of such languages is their agglutinative nature where two or more words combine to form a single word. In such cases, the word lengths grow substantially due to which even for a single character misclassification the whole word is flagged as an error. This again leads to a poor OCR word accuracy.

Detection and correction of OCR errors come under the purview of OCR post-processing. Error detection and correction are two separate terms. Error detection can be described as identifying strings that do not appear in a given word list, dictionary or lexicon [19]. Error correction is a considerably more difficult problem since it involves locating and ranking relevant words to replace the incorrect word. The ranking is done based on a distance metric and the candidate word with the least distance to the error word is selected as the best fit [20]. Such error detection and correction techniques focus on isolated error words and do not take into account the contextual or textual cues. Due to such drawbacks, context-sensitive spell checkers have been widely adopted in OCR post-processing. Niwa et al. [21] proposed an approach where a candidate word is selected based on grammar and vocabulary characteristics around the surrounding error words. One chief disadvantage of such method is that they are unable to correct

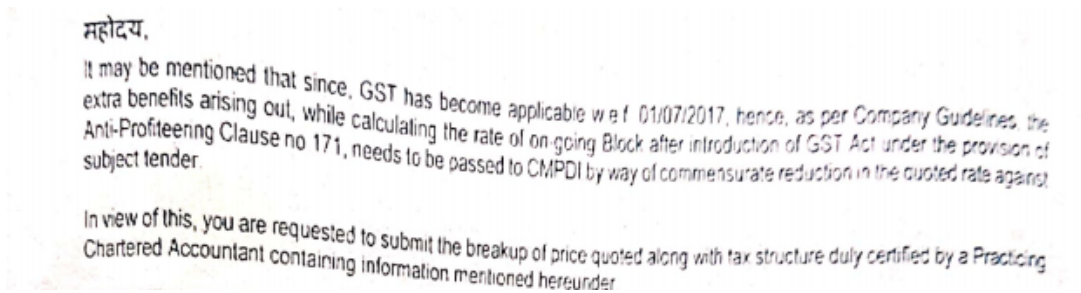


Figure 2.4 Presence of skewed lines and non-uniform illumination in a document image due to incorrect scanning. Such degradations are very common and result in maximum word errors.

real-word errors i.e error words that result in other valid words. Also, such algorithms fail to correct error words which are domain-specific i.e. they depend on the content of the book eg. Nouns, abbreviations, dates etc. Wick et al. [22] proposed to use topic modelling to correct OCR output text where they classify documents based on their content using an unsupervised learning algorithm. This is followed by replacing the error word with a candidate that belongs to the same category as the error word. Bassil and Alwani [23] proposed a context-sensitive error detection and correction method based on the frequency of n-grams for various correction candidates. Using this method they were able to correct both non-words as well as real-word errors. The success of their approach lies in the massive Google Web IT 5-gram dataset mined from various public websites.

2.4.1 Post-processing for Indic Languages

Error detection and correction algorithm depend heavily on the availability of a huge text corpus. Research related to error correction and detection focuses mostly on English or some other Latin languages like German or Spanish. This can be attributed to the availability of large amounts of text data for such languages. However, procuring and maintaining such large corpora for Indic languages is not a feasible option. This is mainly because Indic languages are not as popular as their Latin counterparts. Thus their presence over the web is very limited. Moreover, most of the Indic languages are morphologically rich and agglutinative. New words are often dynamically formed from two or more valid words. Thus, the vocabulary of such languages become extremely large which makes it very difficult for post-processing algorithms to get any meaningful results. Sankaran and Jawahar [24] proposed a binary classifier to detect error words based on n-gram probabilities on Telugu and Malayalam. In [25], the authors develop a feature representation using the frequency of bi-gram and tri-gram for the words and train a classifier to detect error word for Hindi. Similar error detection strategies on inflectional languages have also been attempted in [26, 27].

2.4.2 Post-processing in large document collections

One of the main advantages of a digital library is that the book collection is indexed and catalogued which means, given a query a user can search over the entire collection and retrieve the relevant books where the query might be present. However, this very useful feature is compromised if there are too many error words in the transcribed documents. For example proper nouns, acronyms are often misrecognized. Their replacements do not exist in a dictionary, hence they cannot be replaced automatically. This brings down the quality of retrieval since such words have high recall value. To address this issue, project Gutenberg employed distributed proof-reading [28] where two proof-readers have access to same OCR outputs and they refine them in turns. The Google book project, on the other hand, introduced *Recaptcha* [29] to correct error instances via massive crowdsourcing. The major demerit of the above approaches is that each error word has to be individually corrected. A typical book comprises of 200-300 pages and if each transcribed page supposedly consists of 20-30 error words, then a proof-reader will have to correct 4000-9000 error words per book. This is a huge number considering there are millions of books in a digital library.

Despite the disadvantages, the large volume of pages can help us to improve the recognition rates of OCRs. Firstly, books are usually type-set in a single font which implies that we can retrain our OCR on the initial few pages so that it can generalize well to the rest of the collection [30]. Secondly, a book can be considered as a single coherent text. Thus, it can provide additional information like language cues and repetition of words both of which can be utilized to boost the performance of the OCR [31]. The above two points have been discussed at length in Chapter 3 and 4 and hopefully it will provide with a better understanding.

2.5 Machine Learning in limited labels setting

In the prior sections, we focused on our OCR architecture and discussed the existing techniques for OCR post-processing. However, in the following sections, we look at some of the machine learning algorithms that are used when enough labelled data is not available. We feel that this is important as it will better acquaint the readers with the complexities involved in a limited label setting and the practices involved to counter such challenges. We also feel that it will lay a foundation for the various approaches proposed in this thesis.

2.5.1 Unsupervised learning

Unsupervised learning, as the name suggests refers to those algorithms that learn without any external supervision. Clustering algorithms like k-means [32], GMM, MST etc. learn to identify pattern in the data and group those elements which share similarity on an high dimensional plane. The underlying mechanism of such algorithms involves the construction of an affinity matrix between each pair of ele-

ments present in the data based on some distance metric such as euclidean distance or cosine similarity. Based on the affinity matrix and a user-defined threshold, elements in the dataset which lie very close to each other are grouped into a cluster. In Chapter 3 we make use of k-means to group erroneous predictions of similar word images based on image features extracted from a deep convolutional neural network and correct them in batches. We show that although the algorithm is very simple yet it is effective in reducing the correction cost significantly. Despite their effectiveness, k-means or GMM do not scale well as the size of data increases. This happens because in order to create the affinity matrix the clustering algorithms need to compare each element present in the dataset with every other element. This becomes a bottleneck in our proposed approach since we are concerned with error correction efforts in large document collections. To counter this drawback of clustering algorithms we use LSH which belongs to the family of approximate nearest neighbour algorithms. LSH doesn't compare the distance between all the points and instead takes an approximate approach towards clustering. This brings down the computational time drastically down and proves an effective method in our batch correction pipeline.

2.5.2 Transfer learning

Deep learning (DL) models often fail to generalize to new data whose distribution is different from the data on which it was trained. This can be attributed to the domain shift between source and target data. This poses a serious drawback since failure to generalize to new task suggests that for each task we must have a separate model. We know that DL based models require large amounts of labelled data to learn meaningful representations. However, it is not always feasible to obtain labelled data pairs owing to high annotation cost. This forms the motivation of transfer learning which seeks to leverage the knowledge of a pre-trained models to achieve comparable results on a new target task. Transfer learning has been widely adopted in the domain of computer vision. Image representations learned with CNN on a large scale annotated data can be transferred to other visual tasks that have limited annotated training data [33]. Models pre-trained on the Imagenet and MS-COCO dataset, [34, 35, 36] have achieved superior results on tasks like object detection, classification and image segmentation as compared to models that have been trained from scratch for a specific task.

2.5.2.1 Semi-supervised Learning

Although transfer learning techniques like finetuning work reasonably well when we want to adapt an existing deep learning model to a new task yet it requires a sizable amount of data pertaining to the target task to be annotated. Additionally finetuning does not utilise the vast amounts of unlabelled data which is relatively cheap and easy to access. In the past few years, there has been an increased interest among the research community to develop techniques which can utilise the huge amounts of unlabeled data along with labelled data to learn better representations. In this section, we provide a brief overview of semi-supervised approaches that use both labelled and unlabeled examples for learning.

2.5.2.2 Self-training

Self-training [37, 38] was one of the earliest attempts to use unlabelled data to boost model performance. Self-training comprises of two stages: Initially, the model is trained on a limited amount of labelled data using a supervised method. In the next stage, the learned model is used to predict the labels of unlabeled data points. Finally, the model is trained on both the labelled and unlabeled data where the predictions of unlabeled data are treated as target labels. One of the major concerns of self-training is that initial trained model might predict a significant amount of unlabeled data erroneously. This might bring down the performance of a model while training rather than improving. Care should be taken to minimize the number of noisy predictions in the training set. One way to do so would be to screen the predictions effectively and include only those predictions on which the model is highly confident. Despite the screening measure, some noisy predictions still manage to creep into the training data, which hinders the learning of the ML algorithm. Most works in this domain discuss providing perturbations to the input data or model as a way to overcome confirmation bias. In our work, we use a form of linear perturbation known as mixup and as well as some other regularization method to limit the effect of noisy predictions in our training data. We discuss more on this in Chapter 4

2.5.2.3 Label propagation

Label propagation is another technique that comes under semi-supervised learning. The main idea behind label propagation is to construct a graph using both labelled data as well as unlabeled data which is then used to propagate the labels from labelled data points to unlabeled data points using cluster assumptions. Label propagation is transductive which means that both labelled and unlabeled examples are used to predict the labels of unlabeled examples. It comprises of two steps: construction of a graph and inference. In the graph construction stage, data points of both labelled and unlabeled data form the nodes while the edges represent the similarity between the data points. Larger edge weights indicate higher similarity between the data points and vice-versa. The most common technique to create a graph is using a clustering algorithm such as kNN where the edge weights are obtained using as a euclidean based distance function. In the inference stage, the labels from the labelled data are propagated to their nearest unlabeled data points along with a certainty score which is simply the Euclidean distance from the nearest labelled data point.

Chapter 3

A Low Cost Correction Approach for improving OCR Word Accuracy in a Large Document Collection

In this chapter, we summarize our efforts towards enhancement of word accuracy of English and Hindi OCR documents in a large collection. We look at some of the existing methods in OCR error correction and motivate towards our proposed solution of grouping similar error words and correcting them in one go. To this effect we introduce two variants of batch error correction: Automated and Human aided and discuss the advantages as well as disadvantages of each method. We demonstrate the effectiveness of our proposed solution empirically which achieves over 70% reduction in cost compared to the naive human-centric approach of correcting one error at a time.

3.1 Introduction

The past decade witnessed a growing interest towards the creation of huge digital libraries by digitizing books [4, 39, 3]. One of the crucial steps towards digitization involves the recognition and reconstruction of document image collection(s) using an OCR. The recognition module in the context of digitizing collections such as books, could be considerably different from that of recognizing a single document image [40, 30]. In this chapter, we extend this idea to error correction in document image collections. Often the recognition module of an OCR has an automatic error correction module embedded. This may be using a dictionary or a statistical language model (SLM). However, many applications need further improvement in accuracy. This demands a human intervention for removing these errors. In this chapter, we propose enhancements to the naive human correction approach which reduces the cost for human expert review by more than 70%. Our work is guided by the following two insights. First, the OCR module makes errors consistently. Word images with similar noise drawn from the same type of document leads to same kind of errors. We demonstrate this in Figure 3.3 where instances of the same word images, drawn from a document collection are misclassified consistently by an OCR. Secondly, books/document collection have a finite set of unique words and many of them are repetitions. These may include named entities and domain specific words which are mostly unknown to the error detection

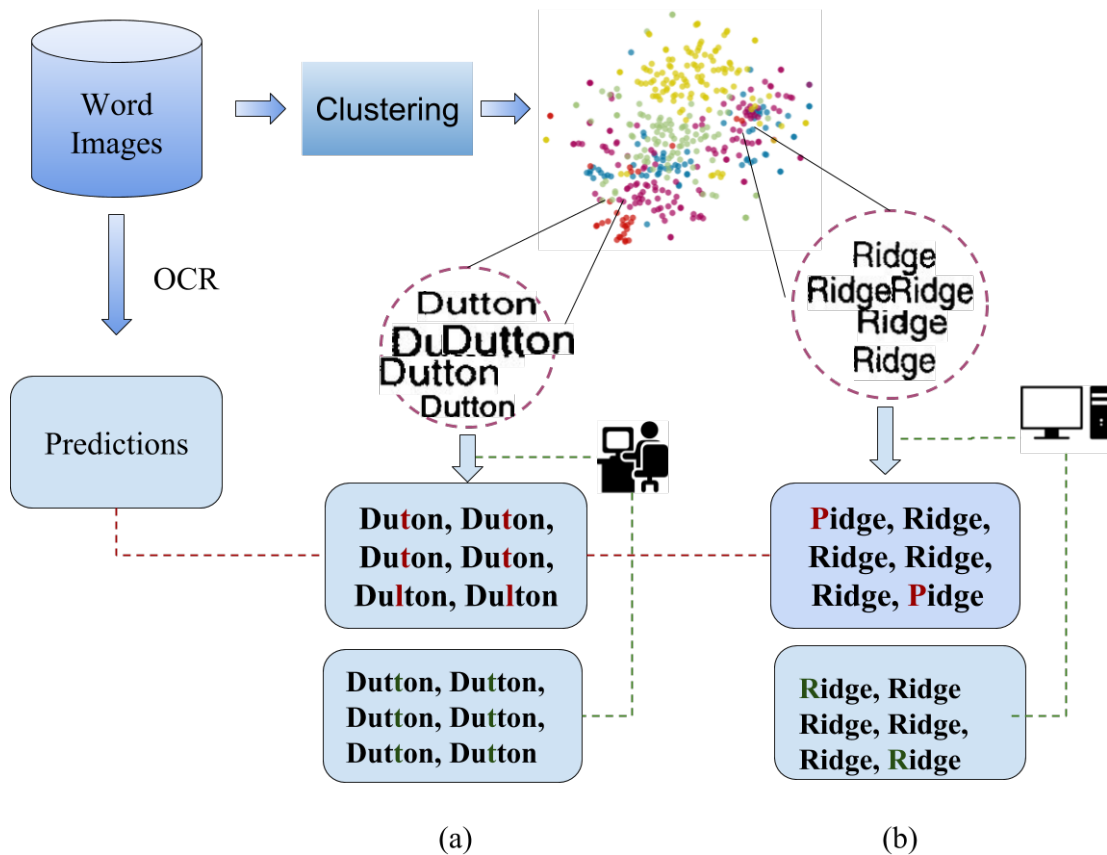


Figure 3.1 The proposed pipeline for batch correction process where the error instances are clustered and corrected in one go. For a group of error instances, the correct label is chosen and applied. The correct label can be either chosen by a human annotator (a) or generated automatically (b).

module. This is further demonstrated in Figure 3.2 where we show that a subset of words in a collection occurs very frequently. A small set of words can cover more than 50% of the total words in the collection. Under this setting, grouping based on image features or similarity in the predictions of the OCR can provide cues for automatic correction or aide the human expert for manual correction.

We model the problem of word error correction as batch correction where the human expert reviews and corrects errors in batches, than in isolation. Figure 3.1 presents an overview of our proposed batch correction scheme. Word image-prediction pairs extracted from a collection of documents form groups based on their image and text similarity. In case such a group is recognized incorrectly by the OCR, only one instance from the group needs to be corrected which is then propagated to the rest of the group elements. Thus, correction needs to be made only once which reduces the cost of correction drastically. The correction can either be automatic or with the help of a human expert. We discuss both these correction processes in detail later in this chapter. The major contributions of this work are:

- We demonstrate how clustering can induce an automatic correction and reduce the manual effort in reducing the OCR errors.

- We empirically validate our approach on multiple languages, multiple OCRs, and on a collection of 100 books.

3.1.1 Related Work

Conventional approaches to error detection and correction reduce to finding the closest match for an invalid word in a known vocabulary [19, 41]. Bassil and Alwani [41] put forth one of the first works which explored in detail OCR post-processing methods. They consider three modes of correction. In the simplest of approaches, corrections could be performed manually by a human expert. Next, a dictionary-based method similar to what modern day word processors are equipped with was proposed. A possible correction is suggested once an error word was detected. This is accomplished by finding a word in the dictionary with minimum edit distance to the error word which becomes the correction proposal. Dictionary-based approaches could not capture errors in the grammar where words were correct according to the dictionary, but not in the surrounding context. Ability to correct such mismatches was attempted by bringing Statistical Language Models (SLM) using larger language context [23, 42]. The SLMs do not work well for many languages which lack large text corpus to build the language models. Also, they run into issues when newer out-of language words (such as a technical word from a foreign language) come in books. Smith [43] argues that, unless carefully applied, a language model can do more harm than good. Hence it becomes necessary to review the results of a conventional OCR system by bringing a human in the loop for the digital reproduction of a book.

To involve human in the loop, projects as early as Project Gutenberg [4] introduced distributed proofreading [28] approaches. Two proofreaders, having access to a book, refine its OCR outputs in turns. A demerit, in this case, is that the entire book has to be visited for proofreading. [29] suggested *ReCaptcha* and report the use of crowd-sourcing to transcribe word images. While the corrections are made only in the case of suspected errors, the efforts ignore the possibility of grouping similar misrecognized images and propagating the correct label to each instance in one go. Observing OCR errors to be highly correlated, [31] proposes a low-cost method to improve the required human-hours needed for correction using clustering. They group OCR outputs first, followed by finding subgroups using the word-image similarities. A similar method based on word image clustering had been shown [44] to be effective in creating index over a collection of documents. However the above two approaches, assume the clusters to be completely homogeneous and thus fail to address cases where the clusters might contain more than one label. [45, 46] note that enabling information retrieval in document image databases is hampered by errors in the OCR outputs [24]. This is very important for the effective use of digital library for books through large scale digitization which include Project Gutenberg [4], Digital Library of India [39] and Google Books [3]. One of the main objectives of such projects is to provide content level access (enable search and retrieval) over the entire digitized collection. Past works turn to humans for correcting the last array of errors left in the pipeline post recognition [29, 31]. All these leave scope for improvement in the space of error correction, especially addressing challenges while scaling up the number of books. Our work is motivated by the works such as [47, 48, 31] that cluster word images

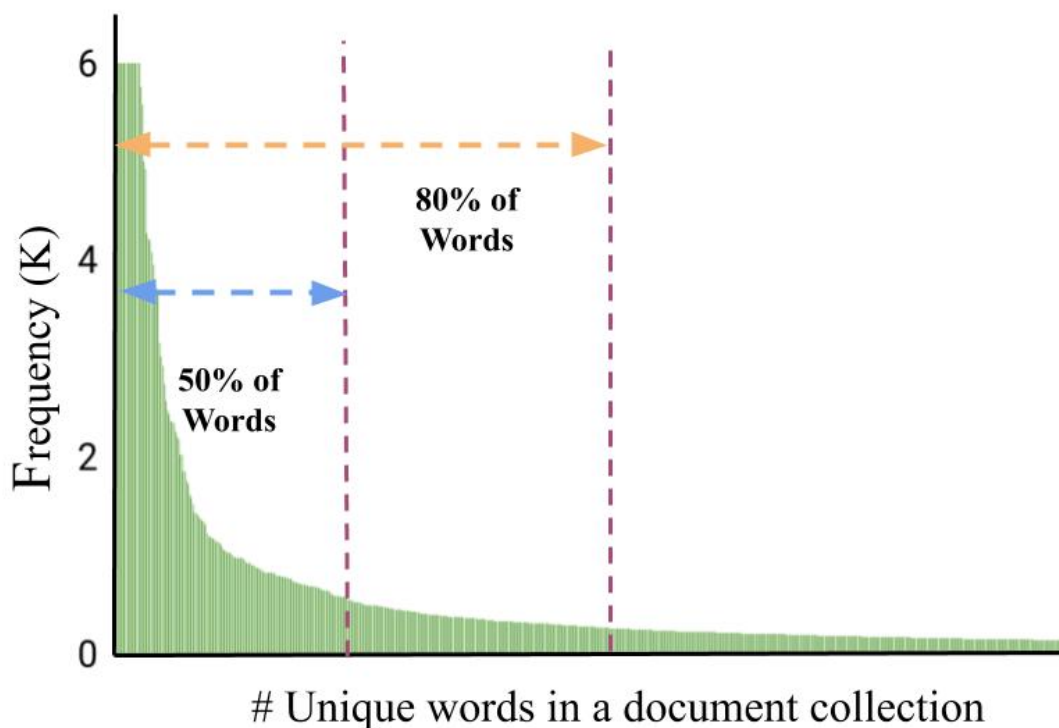


Figure 3.2 The frequency of unique words in a collection of documents. A subset of words in the collection vocabulary have a very high frequency and accounting for 50% of the words present in the collection. Thus it is safe to assume that if errors occurring in this subset are grouped and corrected in a batch, it can lead to a significant reduction in correction cost (Zipf's Law [2]).

to improve the efficiency and accuracy. In this work, we group the erroneous predictions, and present them to a human editor. The human editor then decides the label for the cluster and also the components (elements present in a cluster) to which the label shall be assigned to. The instances where the cluster label do not match the content of the word image are addressed separately by the editor. This mitigates the propagation of errors for clusters that are not homogeneous.

3.2 Cost Effective Correction

In this section we formulate the problem of error correction and propose two strategies of batch correction.

3.2.1 Types of Errors

Recognition modules of OCR systems operate at a character or word level resulting in transcribing word-images into a textual string. Errors in such a setup are inevitable and the cost of manual correction

✓ manner manner	✓ manner manner	✓ manner manner	✓ manner manner	✓ manner manner	✓ manner manner	✓ manner manner
✓ features features	✓ features, features,	✓ features features	✓ features features	✗ feameres features	✗ feameres, features,	✗ feameres features
✓ show show	✓ show show	✓ show show	✓ show show	✗ slow show	✗ slow show	✗ slow show
✓ Juliet Juliet	✓ Juliet Juliet	✗ Juiiet Juliet	✗ Juiiet Juliet	✗ Juiiet Juliet	✗ Iuliet Juliet	✗ Iuliet Juliet
✓ pleasure pleasure	✓ pleasure pleasure	✓ pleasure pleasure	✗ plasure pleasure	✓ pleasure pleasure	✗ plasure pleasure	✓ pleasure pleasure

Figure 3.3 Consistent errors generated by OCR for a given document collection. Each row represents different images for the same word and its corresponding OCR prediction in the green text box. We can observe that for similar degradations, the OCR outputs similar error patterns.

is significantly high. Since it is practically impossible to verify each word manually, we propose to have an independent error detection mechanism operating on the OCR predictions. Assuming that such a system has a low False Negative Rate, only instances where the OCR prediction is not agreed upon by the error detection pipeline, which we denote hereafter as *error instances*, would then need to be corrected. We assume that the errors are detected with a dictionary or an appropriate error detection module. One can categorize the agreement between the recognition module and the error detector into four categories:

1. Error False Positives (EFP): Set of words that are falsely flagged as error by the detection module since they do not exist in the dictionary, such as out of vocabulary (OOV).
2. Error True Positives (ETP): Errors of the OCR which are correctly detected by the error detection module.
3. Recognizer False Negatives (RFN): Words exist in the dictionary but are not the correct transcriptions of the word image.
4. True Negatives (TN): Recognizer correctly predicts word image, and the detection module is in agreement.

Note that the words in TN after error detection are correct words and nothing needs to be done. The words in RFN cannot be detected as an error in isolation. Their correction needs larger language context and is out of scope of this chapter. As far as the error correction is concerned, we would like to take human help or automatically correct the words categorized as ETP.

Our objective is to further reduce the word error rate of an OCR. We exploit the fact that OCR system is prone to make systematic errors. Due to the nature of the prediction, multiple instances of the same word could be misclassified to the same wrong label. We propose a grouping of such misclassifications in a collection of documents which enable correcting these multiple errors in one go.

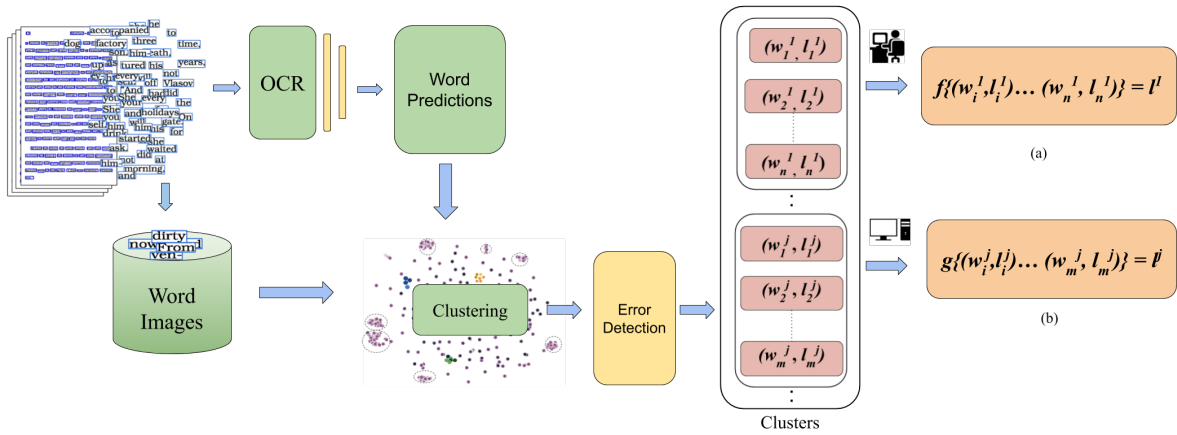


Figure 3.4 Pipeline of proposed batch correction approach. Given word images $(w_i \dots w_n)$ and its corresponding OCR predictions $(l_i \dots l_n)$ we form clusters. Next, the clusters containing error instances are sent for correction. We employ two forms of correction approaches which are shown in (a) when the human editor decides the label for a cluster and in (b) when the cluster label is generated automatically.

3.2.2 Evaluation of Error Correction Effort

We propose a cost based evaluation to demonstrate the efficacy of our method. To this end, we first enumerate all possible edit actions a human in the loop has available and associate a cost with each action.

- We define a verification cost C_v for the case where the reviewer just has to verify an already correct prediction to be a valid word.
- We define average word typing cost C_t for cases where corrections have to be fully typed out.
- For cases where a dictionary provides correction proposals in a drop-down fashion, we define a cost C_d .

For a naive correction process (process where no batching is involved), the editor will have to type out corrections in ETP and verify a word wrongly classified in EFP. The total cost involved turns out to be $C_1 = |\text{ETP}| \cdot C_t + |\text{EFP}| \cdot C_v$. (Here, $|X|$ denotes the cardinality of set X .) We denote this method hereafter as *Typing*. If for some error instances, the editor has an additional option to select from a set of correction proposals, the cost reduces to $C_2 = |\text{ETP}_t| \cdot C_t + |\text{ETP}_d| \cdot C_d + |\text{EFP}| \cdot C_v$ such that ETP_t , ETP_d forms a partition of ETP. Here ETP_t refers to the error true positives which can only be corrected via *Typing* whereas ETP_d refers to the error true positives that can be corrected by choosing the correct suggestion from the set of correction proposals. This method is denoted as *Typing+Selection* hereafter.

3.2.3 Overview of the method

We hypothesize that correcting similar instances in EFP and ETP together can make digitization efforts more efficient. As mentioned above, we propose an approach that groups error instances together, based on some similarity metric and propagate the correction of one of these to the rest of the group. Correction candidate for a group of error words can either be fully automated or done with human aid. We discuss both the propositions in detail later in this section. In the ideal case, word images with same ground truth will be grouped together, and the ability to correct them in one go would provide an efficient way for humans in the loop to correct large document collections. If we could group the error instances based on their ground truths as $C_1, C_2 \dots C_{|V|}$, each of these groups could be corrected in just one action from editor leading to a cost of $V_t C_t + V_d C_d + V_v C_v$ such that $V_t + V_d + V_v = |V|$. Here V_t, V_d and V_v are numbers of clusters requiring typing, selection from dictionary and verification respectively.

Our proposed model for error correction is presented in Figure 3.4. The document images, segmented at word level go through the OCR pipeline which assigns them labels/predictions. The word images and their corresponding predictions are subsequently sent through a clustering pipeline, which groups the word images based on their image and text similarity. We discuss the clustering pipeline along with the features on which the clustering is performed in Section 3.2.3. Next we perform an error detection on the components of each cluster and identify those clusters in which error instances occur. Only those clusters which contain error instances are sent for either of the two correction techniques-automated or human aided, which are discussed below.

Automated approach For a given cluster containing word images and their corresponding OCR predictions, the most frequent prediction is chosen to be the representative of the whole cluster and its label is propagated to the remaining cluster elements. Two scenarios arise out of such a setting. For a given cluster, the number of correct predictions can either be (i) more or (ii) less than the incorrect predictions. This leads to two situations.

In case (i), words appearing in ETP get corrected automatically without any further manual corrective action other than verification. In the case (ii), words appearing in EFP (proper nouns, acronyms, technical keyword, etc. which are flagged as error due to their absence in the dictionary/lexicon) get corrected without much cost. However, this does not hold true for clusters containing ETP where even the few correct predictions that might be present in the cluster end up being assigned the wrong label (since incorrect predictions are more in number). Thus a human editor is required to verify the assigned label with the actual word image for every erroneous prediction and make keyboard entries wherever necessary. This leads to an added correction cost.

Human aided approach We allow a human editor to pick the representative of the cluster. This reduces the cost by eliminating the chances of error propagation which arise when labels are generated automatically. However, this also mandates that a human editor be present throughout the correction

process. In case of ETP, the editor can enter the correction once and the correction is propagated to all matching images. Our method here reduces the cognitive load for the human, thereby improving efficiency.

In the above two approaches we consider the clusters to be completely homogeneous. Clusters containing impurities and the relevant correction approach is discussed later in the chapter.

3.3 Grouping Error Words

In this section, we provide the details of our approach for grouping error words together. As discussed earlier, this significantly reduces the human cost.

3.3.1 Features and Clustering

For every error instance, we have two types of features for use in clustering: text predictions of OCR and features from word-image.

Image Features We use the pre-final layer representations from deep neural networks trained to classify word-images. Such representations capture the discriminatory information between different word-images and have demonstrated success in embedding similar images together [49]. The activation for an image can be considered as a compact representation in a continuous space. For clustering the above features, we employ the k-means [32] algorithm.

The number of clusters k is set to number of unique words in a collection. The k-means algorithm has a time complexity of $O(n^2)$ where n is the number of error instances detected by our pipeline.

Text Features For text features we propose using the word predictions of the OCR. A natural distance measure for such features is the edit distance, which has been found to be of significant help for error detection in past work.

However, approaches like k-means are ill-suited to the discrete nature of these features and our distance measure. Therefore, we propose using a Minimum Spanning Tree (MST) based approach [32] using pairwise edit-distance to cluster variants of text predictions. This could also group consistent errors which comprise of error instances where the (1) Prediction is right but error detection is in disagreement. (2) for the same kind of word-image (different word images of same textual word) OCR consistently give the same erroneous prediction due to bias in training data.

For clustering, we consider the predictions as vertices of a weighted undirected graph, and the pairwise edit-distance between two vertices form the edge weights. Distances between vertices are scaled to $[0, 1]$. A MST is constructed and edges with weights greater than a threshold are discarded, which results in a forest where each connected component forms a cluster.

Image Features and Text Word images with high visual similarity but having different text content can be grouped into the same cluster since they might be close to each other in the image feature

space. This leads to fragmentation or formation of impure cluster. Assuming one true label per cluster can induce an additional cost of correcting word instances whose ground truth is different from the assigned label. To address the intra-cluster variability we further partition each cluster into sub-clusters by leveraging the textual transcription of each word image such that words that lie within a predefined edit distance, can be grouped into the same sub-cluster.

3.3.2 Practical Issues in Clustering Algorithms

In a simpler first approach over a fewer number of books, we use k-means and MST based clustering algorithm to group error instances together. While the two algorithms work well for a fewer number of books, they don't scale to much larger book collections. We address this by using a Locality Sensitive Hashing (LSH) based nearest neighbour computation [50] in our clustering pipeline. We discuss in detail the algorithms and their suitability below.

Degradations in print, paper or both over time are prevalent in older documents. Font styles and variations different from the OCR's training distribution used by a common publishing system across these books could be similar in the image space. Similar noise in the images like the cuts and merges lead to consistent errors in OCR. This prior domain knowledge can be incorporated and taken advantage of while clustering.

Under these circumstances, we find LSH well suited for scaling up correction in our problem setting. LSH tends to approximate the nearest neighbour search in a way such that items which are similar are hashed into the same 'bucket'. Consistency in noise leads to similar hashes for features from images with similar content. Search space is now limited to the bucket of word-images for which hash matches the query image. This makes the process orders faster.

3.4 Dataset and Evaluation Protocols

We experiment on two datasets (both are collection of books). First one fully annotated (FA) and the other one partly annotated (PA). Annotation here imply that we have an error free transcription that allows us to validate whether our corrections are valid or not. This is done by having an error free transcription of the book.

Table 3.1 summarizes these dataset.

scale	Language	#books	#pages	#words	# unique
FA	English	19	2417	0.73M	30K
	Hindi	32	4287	1.20M	63K
PA	Hindi				
	- Annotated	50	200	30K	6K
	- Unannotated	100	25K	5M*	80K*

Table 3.1 Details of the books used in our work. Here FA refers to the fully annotated books whereas PA refers to the partially annotated books.

Method	English						Hindi					
	Automated			Human			Automated			Human		
	Type	Type + Select		Type	Type + Select		Type	Type + Select		Type	Type + Select	
-	S	G	-	S	G	-	S	G	-	S	G	
k-means(I)	1.13	0.87	0.69	0.68	0.52	0.37	1.01	0.71	0.64	0.49	0.36	0.23
LSH(I)	0.93	0.73	0.69	0.28	0.23	0.22	0.94	0.66	0.65	0.16	0.13	0.13
MST(T)	1.0	0.74	0.69	0.19	0.18	0.18	1.0	0.69	0.68	0.14	0.13	0.13
k-means(I) + MST(T)	1.0	0.85	0.65	0.60	0.45	0.32	0.96	0.68	0.63	0.28	0.21	0.19
LSH(I) + MST(T)	0.94	0.73	0.68	0.28	0.23	0.22	0.94	0.66	0.65	0.15	0.12	0.12

Table 3.2 Evaluation of costs of each approach proposed in this paper. The numbers reported are relative to Full Typing method. We observe a decrease in cost as we go left to right for each clustering approach for books of a given language. ‘I’ stands for image features, and ‘T’ stands for prediction text.

Fully Annotated(FA) This annotated dataset comprises of 19 books in English and 32 books in Hindi. Pages from the books are segmented at a word level and annotated by humans. Five books are set aside from each of the languages to train the OCR while rest of the books are used for testing and further batch correction experiments.

Partially Annotated(PA) In order to demonstrate the scalability of our approach, we run our experiments on a larger collection containing 100 Hindi books. Most of these books were printed decades ago, resulting in degradation in quality of pages. The collection consists of approximately 25K pages with around 5M words. A small subset of 200 pages across 50 books are annotated and set aside as test set

3.4.1 Evaluation Protocol

Our primary objective of evaluation is to find the effectiveness our proposed batch correction method, as compared to the naive approach where each error is corrected individually.

To that effect we use units of seconds that a human expert is required to put in for each method. We also analyze the gains across the various clustering approaches as well as different features used for clustering. Having fully annotated (FA) ground-truth information allows to estimate the costs.

In partially annotated (PA), we evaluate the performance of our approach on a large collection of 25K pages. Though we use the entire 25K pages for clustering, performance is estimated using the 200 annotated pages, randomly sampled from the books. We hypothesize that the increase in word accuracy of the OCR translates to a reduction in correction cost. Also, since the subset of pages used in evaluation belong to the same pool of books which the larger clustering algorithm is run on, it is reasonable to assume that decrease in cost during evaluation is indicative of a decrease in the whole collection.

3.5 Experiments, Results and Discussion

In this section we present the implementation aspects of the batch correction model and the results on the two datasets.

3.5.1 Experimental Setup

The OCR we use to recognize the cropped word images follows CRNN style hybrid image to text transcription model first proposed by [51] in their work for scene-text recognition. We also have an error detection module for verifying the accuracy of these predictions, implemented using a dictionary. We trained two OCRs – one for each language. For training, we set aside 5 books each from English and Hindi book datasets respectively. The English language OCR was trained on word images from nearly 600 pages (~160K words) while the Hindi language OCR was trained on word images from approximately 650 pages (~180K words).

The CNN based feature extractor for clustering word images is adopted from [49]. The network was initially trained on synthetic handwritten word images and later fine-tuned on a real-world corpus. Real data used in training is the same 160K word images which are used for training the OCR. The segmented word-images are fed to the network and the pre-final layer activations are used as features for clustering.

Error detection is performed using a dictionary. An instance is determined to be erroneous if its OCR prediction is not present in the dictionary. To suggest corrections for an error instance, the dictionary requires a reasonably good vocabulary. We generate a base dictionary by using Wikipedia dumps for the respective language. For each book while testing, we enrich the corresponding base dictionary further using ground truths of books used for training but not the ones we are testing. We use two variants of this dictionary - one *Static* (S) and the other *Growing* (G). The *Growing* allows for addition of new words to dictionary, like how modern word processors do. In our grouped correction scenario same words could be scattered across clusters and *Growing* dictionary speeds up correction by not having to type the words that are already corrected.

3.5.2 Cluster Impurity

One of the limitations of clustering algorithms like k-means or MST is their inability to form perfect, homogeneous clusters. Despite our efforts in fusing image and text features together in order to minimize the impurities, outliers manage to creep into the clusters. This is shown in Figure 3.5. Cluster impurity poses a serious drawback in our error correction pipeline. Up until now we considered our clusters to be homogeneous and formulated our cost accordingly. However, in practice this can lead to wrong cost estimation. For automated approach, cluster impurity can lead to assignment of labels to instances which do not share the same ground truth. Thus an annotator needs to revisit each cluster and correct all unwarranted cluster assignments.

Romeo ✓	Romeo ✓	Romeo ✓	Romeo ✓	Romeo ✓	Romeo ✓	Romeo ✓
honest ✓	choicest ✗	honest ✓	honest ✓	honest ✓	honest ✓	honest ✓
Capulet ✓	Capulet ✓	Capulet ✓	Capulet ✓	Capulet ✓	Capulet ✓	Capulet ✓
that ✓	that ✓	that ✓	that ✓	hat ✗	that ✓	that ✓
Fool ✓	FOOL ✓	fool ✓	foot ✗	fool ✓	food ✗	food ✗
complete ✓	complete ✓	contemplate ✗	someplace ✗	someplace ✗	contemplative ✗	complete ✓

Figure 3.5 Qualitative results of k-means + MST clustering on English dataset. Images, relevant to the cluster are marked correct while the false positives are crossed out.

For human in the loop, we let the human assign labels to the cluster components. A human can correct impure parts of the cluster by visual inspection through *Typing* or *Selection*. Consistent errors can be corrected for a group in this case, unlike in automated approach giving this method an advantage.

3.5.3 Results and Discussions

Our empirical studies compare costs of error correction across different methods/situations. The cost is measured in units of seconds of human effort put into correction. The following values are used for computing the cost in simulations. We assume C_v as 1, C_d as 5 and C_t as 15 seconds. All costs in this work are computed relative to *Typing*. Table 3.3 shows the cost for correction without grouping efforts. We experiment with setups involving no dictionary, static as well as growing dictionaries, restricting the edit actions available accordingly. We find *Typing* + *Selection* outperforms *Typing* and *Growing* outperforms *Static*, as expected.

	Type	Type + Select	
	-	Static	Growing
English	1.000	0.740	0.695
Hindi	1.000	0.686	0.681

Table 3.3 Relative cost of correction with respect to full typing when no batching is involved.

In Table 3.2, we compare the cost of correction when we employ different clustering schemes. Here corrections are performed in batches. Our results are across the two correction approaches - the first which is automated and the second involving a human editor.

The order among relative costs for edit actions and dictionary variants are consistent with the case without batch correction (Table 3.3). Further, we find sequential refinement of clusters using image features and then text-features perform best among different clustering schemes.

For the automated approach, k-means on image features followed by MST on text features achieve the lowest cost for both the languages. When involving human editor in the process, for Hindi, LSH on

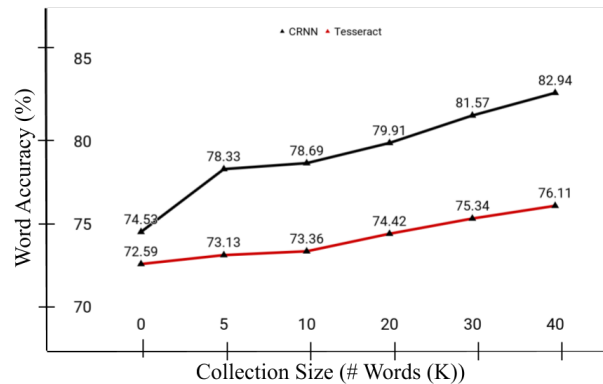


Figure 3.6 Result on the unannotated data. We observe that as the number of words in the collection increases the automated batch correction method’s ability to correct the errored predictions improve which is reflected by the increase in OCR accuracy.

image features and refining with MST works best, while for English data MST on text word predictions seems to achieve the lowest cost (~ 0.13 and ~ 0.2 times the actual typing cost respectively) which is equivalent to more than 70% reduction in human effort.

Correction methods involving human editor consistently outperforms the automated correction approach, even with the former restricted in actions and in dictionary. This can be attributed to the failure of automatic approach in determining the correct label in a cluster which is largely impure.

3.5.4 Results on Large Dataset

We vary the size of the collection and estimate the accuracy on the 200 fully annotated pages. In addition to using the aforementioned hybrid CRNN for text recognition, we also test the effectiveness of our batch correction technique with the publicly available *Tesseract* OCR for Hindi. We observe that the as the collection size increases from 200 to 25K, our batch correction algorithm becomes better at selecting the right candidate and assigning them to the errored predictions. As a result of which the word accuracy post-correction increases systematically. It can be seen from Figure 3.6 that the word accuracy for the dataset improves as the size of collection increases. This implies that for the larger unannotated data, the proposed batch correction method will lead to a better improvement in word accuracy and thus reduction in overall correction cost. We also observe that the gain in word accuracy for our hybrid CRNN-OCR is much more significant than the *Tesseract* OCR. On closer inspection we find that the errors made by the CRNN-OCR mostly fall in the category of EFP i.e. words that were recognized correctly but were flagged down as error by the error detection module. As claimed, these errors can be corrected easily by propagating the most frequent label to all the cluster components. However, this does not seem to be the case with the *Tesseract* OCR where most of the errors fall into the category of ETP. This suggests the the improvement of the OCR word accuracy depends upon the quality and robustness of the existing OCR module. Performance of the traditional methods for error correction does not change with

the size of the collection. Our method scales well to large collections and yields superior performance, making it an ideal candidate for large scale efforts like digital libraries.

3.5.5 Error Analysis

We discuss failure cases of our proposed correction process with a few qualitative examples. For text predictions clustered using MST algorithm, a few error cases are illustrated in Figure 3.7a. The recognition module’s high confusion in predicting numbers and punctuation extends to clustering using text predictions. But there exists strong cues here in the image feature space which can be used to group samples separately.

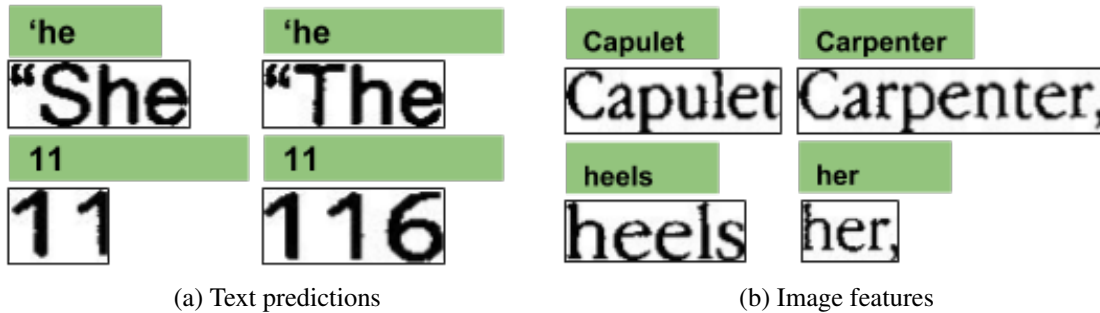


Figure 3.7 Failure cases for clustering on text and image features respectively. Each row in the above figure represents one cluster. The text predictions are depicted in green text box.

Figure 3.7b shows failures in clustering solely using image features. Instances containing ‘Carpenter’ and ‘Capulet’ are grouped into the same cluster although there is a significant difference between their text predictions. Image feature based clustering alone fails to obtain a pure cluster here, but text predictions’ similarity can be used to make clusters more pure. We demonstrate such successful refinement in Figure 3.5. ‘Capulet’ is one such correction proposal, but the entry corresponding to ‘Carpenter’ is no longer associated. Failure cases of the combined clustering approach are indicated in Figure 3.5. Predictions ‘fool’ and ‘food’ are inherently different, but still managed to be clustered together. This is likely due to these being very near in image and text space.

3.6 Summary

In this chapter, we proposed a cost-efficient batch correction scheme for error reduction in OCRs. We achieved this by discovering clusters of similar errors and processing them together. We empirically studied our method in multiple situations and demonstrate the utility. We also demonstrated how the approach scales to larger document collections. Our focus in this chapter was more on enhancing/improving the predictions obtained from an OCR. We do not concern ourselves with improving the OCR model to minimize erroneous predictions. In the next chapter, we take a step back in our OCR pipeline and try to improve our learning algorithm so that it can adapt better to a new dataset without needing too many annotations or labelled data.

Chapter 4

Adapting OCR with Limited amount of Labelled Data

In the previous chapter, we talked about improving the quality of OCR output text from the perspective of post-processing. However, in this chapter, we discuss the problem of minimizing OCR errors from the viewpoint of improving the OCR learning algorithm so that it outputs predictions with lesser number of errors. Generally speaking, the number of OCR errors become more pronounced when an existing OCR fails to generalize to a new dataset. In this work, we hope to curtail this effect via adopting techniques from transfer learning approaches where the knowledge gained by a pre-trained network is leveraged to gain performance improvement on a new task/new dataset. Thus our proposed solution has a flavour of domain adaptation since we focus on improving an existing OCR model on a new target dataset. We also provide solutions when limited annotations are present for the target dataset and show that it achieves comparable results to the traditional transfer learning methods.

4.1 Introduction

At the beginning of the last decade, Deep Learning ushered us into a new era of artificial intelligence. Deep Neural Networks (DNNs) like CNNs [9] and RNNs [10] have shown to learn higher-order abstractions directly from raw data for various machine learning tasks. The need to hand-craft the features for tasks which earlier required domain experts has drastically declined. DNNs have established state-of-the-art results in almost all the computer vision (CV) tasks like image segmentation, object detection, pose estimation etc. Thus, with so much success, it was only natural that DNNs also forayed into one of the oldest computer vision problem of text recognition/OCR.

Motivation One of the crucial steps towards digitizing books is the recognition of the document images using an OCR. More often than not there exists a domain gap between the data on which the OCR was trained and the books on which we want to perform recognition. In the case of digital library creation [3, 4] books that come for digitization will invariably contain variations in their font-style, font-size as well as in their print quality (in case of historical books and manuscripts). Thus an OCR trained on a single source data will invariably fail across such variations. This leads to the inferior performance

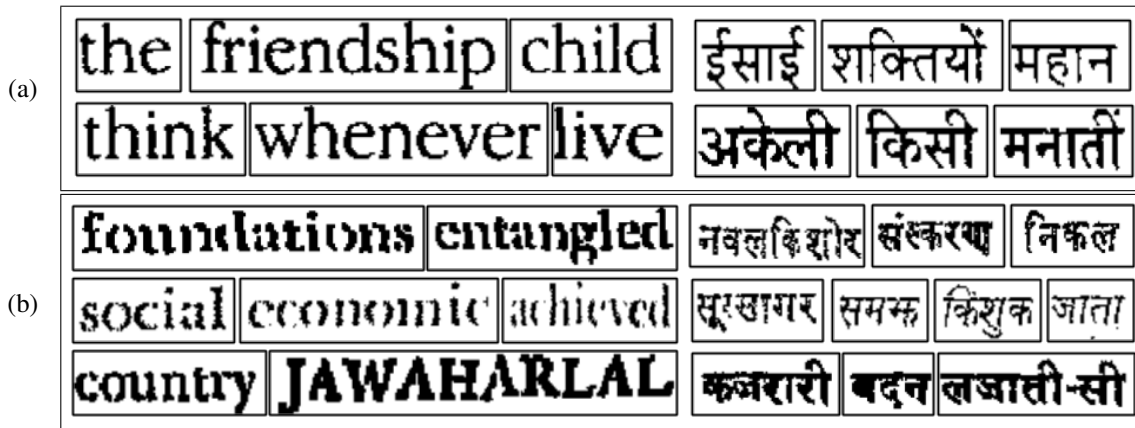


Figure 4.1 (a) Word Images from the collection 1 containing clean-annotated images (b) Word Images from the collection 2 containing partially annotated noisy-degraded images.

of the OCR module resulting in many wrong words. Thus it becomes essential to finetune the OCR model on portions of books that we want to digitize. However, finetuning suffers from its inability to learn from unlabelled data as well as an added cost of annotation. Thus, in this chapter, in addition to finetuning, we also explore a branch of the semi-supervised approach called self-training where the OCR can learn from its own predictions, therefore bypassing the need for the creation of annotated data.

Deep learning in text recognition Traditionally, to recognize text from printed documents, sequential classifiers like Hidden Markov Models (HMMs) and graphical models like Conditional Random Fields (CRFs) have been used. These algorithms were popular since they did not require the line images to be explicitly segmented into words and characters, thus reducing the chances of error. However, due to the inability of such algorithms to retain long-term dependencies, Long Short Term Memory networks (LSTMs) emerged as the de facto choice for such sequential tasks. For recognition of handwritten text, [16] was one of the earlier works to make use of LSTMs. Later it was adopted by [52, 53, 17] for recognition of printed text for Latin and Indic languages. More recently in [1] CNN was used as a feature extractor in the text recognition pipeline, which helped them achieve the state of the art results on various scene text datasets. Taking the same architecture forward, [54, 55] showed that (CRNN) outperformed BLSTM networks on printed documents. In the last few years attention-based encoder-decoder architecture [56, 57] have also been proposed for character sequence prediction.

Learning with limited supervision Although deep learning algorithms have become very popular, much of its success can be attributed to the availability of large quantities of clean annotated data. Collection of such datasets are both laborious and costly and it proves to be the bottleneck in applying such algorithms. A common approach to mitigate the effects of unavailability of clean and annotated data is to leverage the knowledge gained by DNNs on source tasks for whom labeled examples exist and generalize it to target task which suffers from the unavailability of labeled data. Such approaches fall under

the purview of transfer learning which has been widely used in the past especially in Computer Vision (CV) tasks (object detection, classification and segmentation) under a data-constrained setting. Training models from scratch is both resource and time exhaustive. Adapting OCR models by pre-training on ImageNet data and later finetuning on Historical documents has been attempted in the past [58]. However, the improvement in the character recognition rate was not very significant. This may be attributed to the difference in image properties between document images and the natural images found in the ImageNet dataset.

Although, finetuning improves the performance of DNNs reasonably, yet it suffers from the in consequence of having to annotate a sizeable portion of the data which leads to an added cost. Additionally, finetuning also fails to take advantage of the vast amounts of unlabelled data that can be used to enhance the performance of machine learning models. In order to make use of the unannotated data, semi-supervised approaches like pseudo-labeling have become recently popular, where proxy labels are generated for the unannotated images. In [59, 60] the authors showed that by utilizing the unlabeled data in addition to a fraction of the labeled data, they were able to boost the performance of an existing handwriting recognition system (HWR) on a new target dataset. The authors used self-training (discussed in detail in the later sections of this chapter) and were able to achieve a performance gain equivalent to a model which was finetuned on the same dataset but with full annotations.

The key contributions in this chapter are as follows:

- We study the effect of finetuning a pre-trained OCR model on target dataset using a variety of finetuning approaches.
- We also, present a self-training approach for adapting OCR to the target data. We show that by combining simple regularization measures like data augmentation and dropout, we can attain improvement in accuracies close to 11% in the case of English and close to 4% in the case of Hindi dataset with no additional manual annotations.
- We also show that by combining the self-training and finetuning strategy we can outperform models that have been trained exclusively using the finetuning method.
- We empirically support our claims on the dataset for both English and Hindi and show that our proposed approach is language independent.

4.2 Empirical Verification Framework

Dataset Our dataset is divided into two collections 1) Collection 1: which consists of reasonably clean-annotated images on which our OCR is trained. 2) Collection 2: which consists of partially annotated data containing noisy-degraded images. The images in the collection 1 are significantly different in terms of font style and print quality from the document images in collection 2. This can be observed

Collection	Language	Annotation	Purpose	#Pages	#Lines
Collection 1	English	Yes	Training OCR	1000	14K
	Hindi	Yes	Training OCR	4287	92K
Collection 2	English	Yes	Finetuning	50	7K
		Yes	Evaluation	200	9K
		No	Self-training	1100	18K
	Hindi	Yes	Finetuning	250	8K
		Yes	Evaluation	1K	20K
		No	Self-training	5K	100K

Table 4.1 Details of the data used in our work. The table describes the language of the type of collection from which the line images are used. Annotation refers to whether the data split is annotated or not. Also, the column purpose defines the role of each data split.

from Fig 4.1. The various splits for each collection as well as the purpose of each split is described in Table 4.1. We annotate a part of collection 2 and split it into two parts (1) finetuning (2) evaluation. Our main objective is to transfer knowledge from collection 1 to collection 2 with limited supervision.

Evaluation We use the character recognition rate (CRR) and word recognition rate (WRR) metrics to compare the performance of various models. CRR is the ratio of the sum of edit distance between the predicted text (pt) and ground truth (gt) to the sum of total number characters in pt and gt while WRR is defined as the number of words correctly identified, averaged over the total number of words present in the ground truth.

Implementation Details We use a learning rate of 10^{-5} with a step scheduler. Initially we keep the batch size as 32 and the number of epochs as 100. We use early stopping criterion to avoid overfitting and the optimizer used is Adam.

4.3 Finetuning for text recognition

Image representations learned with CNN on a large scale annotated data can be transferred to other visual tasks that have limited annotated training data [33]. Convolution layers pre-trained on a source task adapt better to the target task as compared to a model trained from scratch. There exist several ways to fine-tune a pre-trained model. The most common approach involves training only the last layer while keeping all the layers frozen. Alternatively, another common approach is to use pre-trained weights of a CNN as initialization, thus finetuning all the layers. However, such techniques have gained minimal exposure in the domain of text recognition. In one of the first attempts, [61] reported improvement in

model performance when an existing model is fine-tuned on target data as opposed to training from scratch. We study the following finetuning approaches.

- **Full:** Using a pre-trained model as a mode of initialization and fine-tune all the layers on the target dataset. We train the model on the annotated portion of collection 2 until the loss on validation data stops decreasing.
- **Last:** finetuning only the recurrent layers while keeping the convolution layers frozen.
- **Chain-thaw:** finetuning one layer at a time while keeping the other layers frozen [62].
- **Unfreeze:** Another variation of chain thaw where we sequentially unfreeze the layers and fine-tune each such instance and finally fine-tune the whole network until convergence [63].

We further assess the effect of learning rate schedulers like slanted triangular learning rate (‘Stlr’) [63] and cosine annealing scheduler (‘Cos’) [64] on finetuning.

Method	English		Hindi	
	CRR	WRR	CRR	WRR
Base Model	93.65	83.40	91.63	83.95
Full	97.46	95.88	92.90	86.52
Full + Stlr	98.75	98.22	92.99	86.85
Last	96.16	90.88	92.42	85.27
Chain Thaw	97.96	97.53	93.01	86.75
Unfreeze	98.02	97.75	93.03	86.96
Unfreeze + Cos	98.23	98.01	93.11	87.09
Unfreeze + Stlr	98.79	98.46	93.20	87.40

Table 4.2 Character and Word Recognition results for various finetuning methods. The first row shows the CRR and WRR for the pre-trained model. Subsequent rows contain values for models obtained after finetuning the base model.

4.3.1 Results and Discussions

Table 4.2 shows the result of various finetuning approaches on our English and Hindi datasets. The base model refers to the pre-trained OCR models trained on the clean-annotated source data, which acts as our baseline. Due to the significant difference in the font-style and page image quality between the source and target data, the base model performs poorly as is evident from the results shown. From the experiments, we observe that (1) finetuning the network consistently results in better recognition rates as opposed to training from scratch for both the datasets. (2) finetuning only the recurrent layers (‘Last’) results in under-fitting, particularly on the English dataset. (3) We observe that finetuning the entire network (‘Full’) seems to give us more favorable results on both the datasets. We believe that this is because the target data contains minute nuances in the font style, which the pre-trained

feature extractors are not able to capture well. (4) We also observe that Gradual unfreezing (‘Unfreeze’) approach to finetuning performs the best and is closely followed by ‘Chain-Thaw’. ‘Chain-thaw’ and ‘Unfreeze’ methods work better than the traditional finetuning method since training one layer at a time helps the network to adapt better to the new domain and avoid forgetting. Also, from Figure 4.4a we observe that validation loss quickly converges for ‘Unfreeze’ and ‘Chain-thaw’ techniques with ‘Unfreeze’ attaining the lowest validation error rate. This confirms the effectiveness of the above two approaches. (5) Additionally, learning rate schedulers consistently boost the performance for ‘Full’ and ‘Unfreeze’ methods with ‘Unfreeze + Stlr’ attaining the best overall accuracies. This is in support of our hypothesis that the network learns better, one layer at a time.

4.4 Finetuning across Languages

In addition to the above experiments we also explore the role of languages in finetuning. Languages are strong identifiers for the geographical locations that they originate from. Languages from nearby regions share common traits with each other. This notion stands true with respect to their scripts as well. For example the script Devanagri will have strong correlation with scripts like Gujrati, Rajasthani and Gurumukhi. Similarly scripts from the Southern part of India will share strong characteristics amongst themselves. This motivates us to exploit the aforementioned similarity amongst various languages to the cause of finetuning an OCR. We propose that if a pair of scripts share similar structure and appearance in terms of their constituent alphabets, it will be easier to transfer knowledge between OCR models trained on such language pairs. Besides finetuning we also explore multi-task learning to train our OCR. We propose that it is more advantageous to train a single model on such pairs under the multitask setting instead of individual model for each language. In the multi-task setting we use CRNN model where the weights of the convolution layers are shared between languages of both datasets. However we use separate recurrent layers for each language. The architecture of our multi-task model is shown in Figure 4.2 In the next section, we provide details of our experimental framework which we use to support our hypothesis.

Language	Training	Testing
Tamil	12k	30k
Telugu	65k	30k
Hindi	68k	30k

Table 4.3 Details of dataset used for finetuning across languages

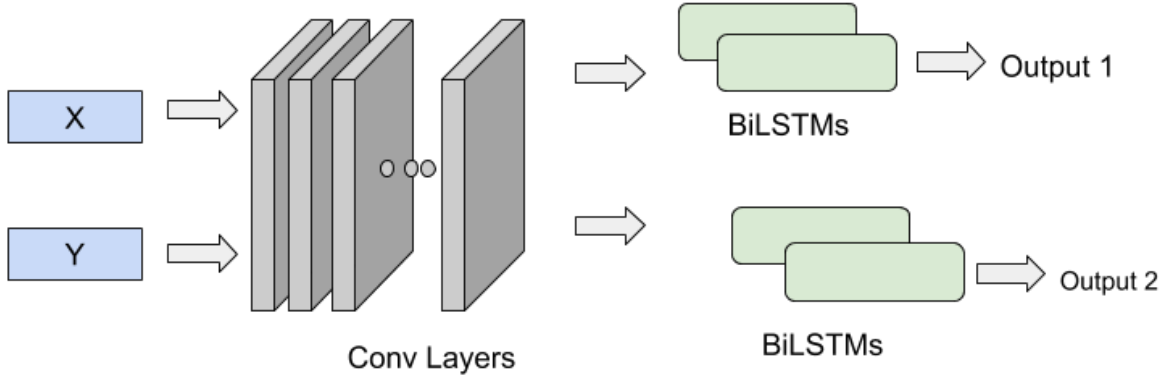


Figure 4.2 Multi-task pipeline for training OCR on two languages simultaneously. Here X and Y refer to dataset belonging to the two languages. The architecture consists of shared convolutional layer weights and separate recurrent layers for each dataset.

Language	Model	CRR	WRR
Tamil	Base Tamil	96.97	82.47
	ft-Telugu	97.26	83.63
	ft-Hindi	96.95	82.52
	mt-Tel-Tam	97.53	85.2
	mt-Hindi-Tam	97.27	83.2
Telugu	Base Telugu	94.71	71.44
	mt-Tel-Tam	95.26	76.76
Hindi	Base Hindi	94.48	83.95
	mt-Hin-Tam	94.47	83.91

Table 4.4 Character and word recognition rates for finetuning and multitask learning on different language pairs

4.5 Dataset

To observe the effect of languages on finetuning we use data from Hindi, Tamil and Telugu. We choose Tamil and Telugu since both the scripts are from Southern part of India and their script characters share similar structure and appearance. In addition we also choose Hindi as a control in our experiments. Hindi uses Devanagiri script which is substantially different from either Tamil or Telugu. By showing a contrast in the finetuning performance between Telugu-Tamil and Hindi-Tamil, we hope to establish that correlation between language is a strong prior for transfer learning in OCRs. Table 4.3 provides details for the dataset used in our experiments. We can observe that the number of line images in our Tamil data is much lesser as compared to Telugu and Hindi. This serves as a motivation for us to improve the OCR accuracy by transferring knowledge from Telugu and Hindi which have substantially more annotated data.

4.6 Results and Discussions

Table 4.4 present the results for our finetuning across languages experiments. To convey results more effective we follow the naming convention listed below:

- **Base X** refers to the model which has been trained exclusively on the data from language X.
- **ft-X** refers to model pre-trained on language X and finetuned on Tamil.
- **mt-X-Y** refers to model trained under multi-task setting on pairs X and Y.

Additionally, the column Language tells us the language on which the model is tested or evaluated. From Table 4.4 we observe that model ft-Telugu outperforms ft-Hindi as well as the Tamil base model which suggest that correlations indeed exist between Tamil and Telugu. Moreover ft-Hindi does not achieve much improvement from the base model. We see the same behavior in multi-task training experiments as well in which model mt-Telugu-Tamil outperforms mt-Hindi-Tamil and it also achieves the best result on Tamil evaluation dataset. It is also interesting to note that by training OCR model on Tamil and Telugu data simultaneously we were able to achieve significant gain on Telugu test data as well using the same mt-Telugu-Tamil model. However, the same cannot be said for training model mt-Hindi-Tamil where the model does not show much improvement on either Tamil or Hindi test data. This provides us with enough evidence that indeed some scripts share common characteristic’s which can be leveraged to improve the performance of OCR on those languages.

4.7 Self-training for text recognition

Self-training is one of the widely used approaches in semi-supervised learning. In this method, we generate the prediction for the unannotated data using a pre-trained model and then use them as pseudo-labels to train a new model [65, 66]. Formally, suppose we have m labeled data (L) and n unlabelled data (U) such that $n \gg m$. Additionally, L and U do not come from the same source which introduces a domain shift. M_0 is a pre-trained model trained on L . M_0 is used to generate predictions for U such that we now have images along with their predictions $[(I_0, y_0^*), (I_1, y_1^*) \dots (I_n, y_n^*)]$. The most confident samples are taken and added to the labelled data L . The model M_0 is then trained on the $n + p$ samples and at the end of the training, we obtain model M_1 . This process is repeated for a fixed number of cycles such that at the end of each cycle, model M_{i+1} is obtained, which is then used to generate pseudo labels on the remaining $n - p$ unlabelled data samples. The process is continued until there are no more unlabelled samples or when there is no improvement in accuracy on the evaluation dataset. Figure 4.3 illustrates the self-training approach. We follow the same procedure for training the model using the self-training strategy on our unlabeled dataset.

Although self-training has shown success in a variety of tasks it has the downside of suffering from confirmation bias [67], where the model becomes over-confident on incorrect pseudo labels thus hindering the model’s ability to learn and rectify its errors. Stuner et al. [59] proposed to use a lexicon

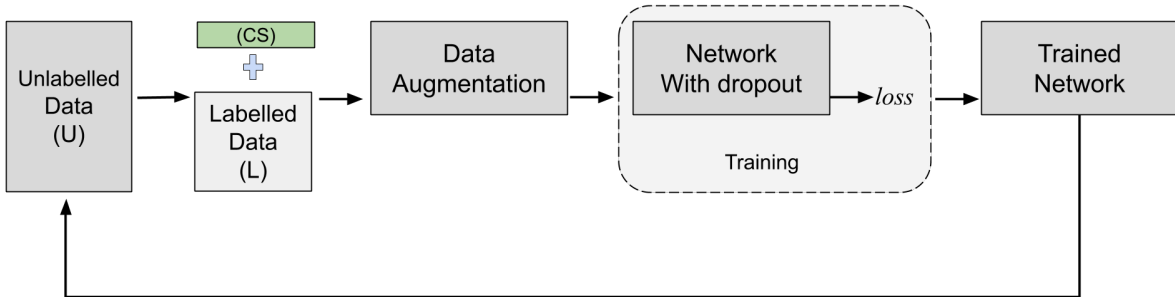


Figure 4.3 A pipeline of our proposed iterative self-training approach. At each iteration model, M_i performs inference and confidence estimation on the unannotated dataset. Top k confident samples is mixed with the labelled data L and the combined samples are noised. The network (M_i) is trained on the combined samples at the end of which we obtain model M_{i+1} . The above procedure is repeated until there is no more improvement in word accuracy on test data.

to verify the correctness of the pseudo-labels thereby avoiding the inclusion of any incorrect labels in the training data. However in the absence of a pre-defined vocabulary (in case of highly inflectional languages like Hindi/Tamil) it becomes essential to carefully regularize the network to alleviate the effects of confirmation bias. We use the recently proposed mixup strategy [68] to augment data which is an effective method for network regularization. Additionally, we also provide perturbations to the network using dropout regularization [69], which further mitigates the confirmation bias and enhances the model’s performance. We show that by applying careful regularization measures to our self-training framework, we can achieve comparable accuracies to the finetuning approach on the Hindi dataset even though no actual labels were used. We also show that by initially training the network on pseudo-labels followed by finetuning on actual labels helped us achieve the best accuracies on both English and Hindi datasets.

4.7.1 Details

Confidence estimation To estimate the network confidence for each unlabelled sample in U , we take the log probability distribution given by the model M_i over the C classes where each class represents a character present in our vocabulary as shown in equation 4.1, where p_i is the probability outputted by the RNN at each time step t . We sort the predictions in decreasing order of their confidence scores and take the top 20% at the beginning of each cycle. However, due to the presence of a domain gap between the source and target data, the network tends to confuse between similar-looking characters resulting in a high probability of being assigned to the wrong character. This leads to predictions containing a considerable number of error words. When the network is trained over such confident but wrong predictions, the errors get reinforced in the subsequent trained models. This further amplifies the errors, which shows that the probability distribution can be a poor estimator for selecting the pseudo labels. To neutralize our dependence on the model’s probability distribution as confidence estimator, we also take

into consideration, the perplexity score of each prediction obtained by a pre-trained language model and finally take a weighted sum over both as shown by equation 4.2.

$$\text{score} = - \sum_{i=1}^t \log(p_i) \quad (4.1)$$

$$\text{score} = -\alpha \sum_{i=1}^t \log(p_i) + (1 - \alpha) \frac{1}{m} \log P(w_1, \dots, w_m) \quad (4.2)$$

where, $\log P(w_1, \dots, w_t)$ is the joint probability of a sentence and α is the weight parameter that we determine empirically. The language model tends to assign a low score to predicted sentences that have error words, which lead to the overall score being low. This leads to sentences with error words to get weeded out from the confident pseudo labels, which in turn helps the model to avoid confirmation bias while training.

Prediction Ensemble Additionally, while computing the maximum likelihood given by model M_i for each sample, we also take into consideration the probability values outputted by the earlier models i.e. $M_0, \dots, M_i - 1$ which is given by equation 4.3 where Z_i is the model prediction at iteration i and z_{j-1} is the average of predictions for models at iteration 0 to $i - 1$. Thus, Z contains a weighted average of outputs of an ensemble of models, with recent models having greater weight than the distant models. In the first cycle of our iterative self-training method, both Z and z are zero since no previous models are available. For this reason, we specify λ to be a ramp-up function to be zero on the first iteration. The idea has been borrowed from [70] where it is used to enforce consistency regularization for semi-supervised learning.

$$Z_i = \lambda Z_i + (1 - \lambda) z_{i-1} \quad (4.3)$$

4.7.2 Regularization

Regularization plays an essential role in our design of the self-training framework. We regularize our network mainly by two ways 1) perturbing the input data and 2) perturbing network. Perturbations to the input data are done by adding Gaussian noise to the input image. In contrast, perturbations to the network are provided by adding a dropout layer, the details of which we discuss in the following sections. Earlier works [71, 72] noted that providing perturbations enforced local smoothness in the decision functions of both labelled and unlabelled data. It has also the added advantage of preventing the model from getting stuck at local minima and avoid overfitting.

Gaussian Noise We multiply the input image with a mask sampled from a binomial distribution. The mask zeros the pixel values at multiple locations, resulting in loss of information. This forces the model to become more robust while making predictions.

Model	Proposed				Baseline			
	English		Hindi		English		Hindi	
	CRR	WRR	CRR	WRR	CRR	WRR	CRR	WRR
VGG + BiLSTMs	96.53	94.01	93.04	87.23	93.65	83.40	91.63	83.95
ResNet18 + BiLSTMs	96.76	94.64	93.22	87.90	95.23	89.71	92.30	85.97

Table 4.5 Comparison of word and character recognition rates of CRNN models between baseline and our self training framework.

Mixup In addition to Gaussian Noise, we also experiment with another type of data augmentation known as mixup [68]. Mixup creates new training samples using a weighted interpolation between two randomly sampled data points $(x_1, y_1), (x_2, y_2)$. where $\lambda \in [0, 1]$ is a random number drawn from a $\beta(\alpha, \alpha)$ distribution. Mixup encourages the model towards linear behavior in between training samples. Additionally, mixup has the property of curbing confirmation bias by enforcing label smoothness by combining y_i and y_j as noted by [73]. This is especially important from the perspective of self-training strategy since we are using predictions of unlabelled images as targets. In such cases, the networks, while training, tend to overfit to its predictions.

$$\hat{x} = \lambda x_i + (1 - \lambda)x_j \quad (4.4)$$

$$\hat{y} = \lambda y_i + (1 - \lambda)y_j \quad (4.5)$$

Weight Dropped LSTM In addition to the above data augmentation techniques, we also apply dropout to the recurrent and fully connected network (FCN). The recurrent layers (Bi-LSTMs) is the most fundamental block in any modern OCR models. Therefore, it only makes sense to regularize them for optimal performance. Dropout is one of the most widely used approaches towards regularizing a neural network. However, naively applying dropout to the hidden state affects the RNN’s ability to retain long-term dependency [74]. We use Weight Dropped LSTMs proposed in [75] which uses DropConnect on the recurrent hidden to hidden weight matrices. We also introduce dropout on the two fully connected layers after each Bi-LSTM layer with dropout probability set to 0.5. The above data augmentation and model perturbation techniques force the model to act as an ensemble which is known to yield better results than a single network in the ensemble.

We also use slanted triangular learning rates (STLR) proposed in [63] for scheduling our learning rates (LR). The authors argue that STLR enables the network to quickly converge to a suitable region in the parametric space at the start and then gradually refine its parameters.

4.7.3 Results and Discussions

Table 4.5 presents the results of our self-training strategy on two kinds of CRNN architectures where the CNN part comes from 1) VGG 2) RESNET18. We observe a significant improvement in word and

Heuristics	English		Hindi	
	CRR	WRR	CRR	WRR
ST	94.22	85.12	92.13	85.41
+ STLR	94.72	86.88	92.17	85.45
+ noise	95.61	91.87	92.26	85.57
+ dropout	95.99	92.57	92.26	85.54
+ mixup	96.48	93.57	92.57	86.23

Table 4.6 The breakdown effect of each regularization heuristic on VGG CRNN model

Cycles	English		Hindi	
	CRR	WRR	CRR	WRR
Cycle 1	96.49	93.88	92.57	86.23
Cycle 2	96.52	93.92	92.95	87.07
Cycle 3	95.52	93.97	93.04	87.23
Cycle 4	95.53	94.01	93.04	87.22

Table 4.8 Character and word recognition rates at the end of each iterative cycle for both English and Hindi datasets

	English		Hindi	
	CRR	WRR	CRR	WRR
ST base	94.22	85.12	92.13	85.41
ST noise	95.61	91.87	92.17	85.47
ST dropout	95.22	89.28	91.91	85.27
ST mixup	96.09	91.73	92.57	86.46

Table 4.7 Ablation study of various refinements

	CRR	WRR
scoring		
prob-dist	96.48	93.45
lang-model	96.01	91.56
weighted score	96.52	93.88

Table 4.9 Character and word recognition rates for different scoring mechanisms on English dataset

character accuracies from the baselines for both the models which shows that the refinements that we suggest are model agnostic and can work under any setting. It is interesting to note that our self-training method on the Hindi base model improves the word recognition rate quite significantly and brings it at par with the ‘Unfreeze + Stlr’ which is the best performing finetuning approach (shown in Table 4.2), even though no actual labels were used to train the network. In the case of English, our proposed approach also performs comparatively well with an improvement of 10% in the WRR and 3% in CRR but lags behind the ‘Unfreeze’ methods. This can be attributed to the difference in the number of training examples between the two datasets with Hindi being far superior in numbers. Thus, we can conclude that the self-training framework benefits from the amount of training data. Additionally, the testing accuracies for each refinement are shown in Table 4.6. By perturbing input images followed by data augmentation using mixup and adding weighted dropout, we systematically improve the recognition rates at both character and word level. Also, it is essential to note that the values are shown in table 4.6 are for models that have been trained for only one cycle of our self-training framework and top samples were generated using sum over log probabilities.

4.7.4 Observations

To study the impact of individual regularization measures on the performance of our self-trained models we add each regularization one at a time. We then train the model with our self-training framework and check the performance for each model on the evaluation dataset. We report the accuracies in Table 4.7. For the above experiments we use VGG16 architecture and we run on only one cycle of

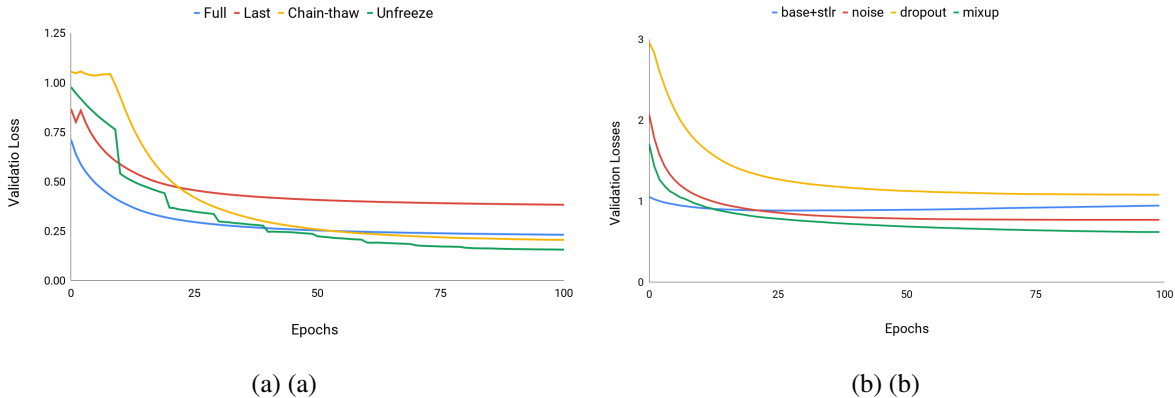


Figure 4.4 (a) shows the Validation curves for different finetuning methods (b) Validation vs. Epochs plots for the self-training method with various regularizers on English dataset.

our iterative self-training framework. From our experiments, we observe that (1) only by corrupting the images with Gaussian noise the self-trained attains better recognition rates compared to the base model. The improvement is rather significant especially in the case of English. (2) Similarly, in the case of dropout, WRR and CRR show an improvement of 4% and 2% respectively. However, we observe that in the case of Hindi, the performance drops. We believe that this happened due to the dropout model under-fitting on the Hindi data and believe that a lower dropout probability will easily fix the issue. We also compare the performance self-trained model with and without the *mixup* data augmentation technique. In this study, no other perturbations were provided. We observe that (4) *mixup* improves the character and word recognition rate on both the datasets, Also, from Figure 4.4b we observe that the validation loss for self-training with mixup strategy converges more quickly than other regularization techniques which demonstrates the effectiveness of the mixup strategy in dealing with the confirmation bias. Additionally, we also study the effect of our iterative self-training strategy and (5) report the gradual increase in accuracy at the end of each self-training cycle in table 4.8. To show the effectiveness of our proposed weighted sum approach for confidence estimation against log probability distribution and sentence probability score. From Table 4.9 we observe that the proposed weighted scoring achieves the best accuracies. Since no pre-trained language model was available for the Hindi language, this set of experiments were performed only on the English data. Using only the probability scores from the language model results in an inferior performance. This happens because every language models suffer from an inherent bias towards the dataset on which it was trained. Also, sentences containing proper nouns and abbreviations were given a low score in spite of being correct.

4.8 Hybrid Approach: Self-training + finetuning

Now, we combine both finetuning and the proposed self-training framework to further improve the performance of our OCR. The success of a finetuning depends on how well weights of the pre-trained

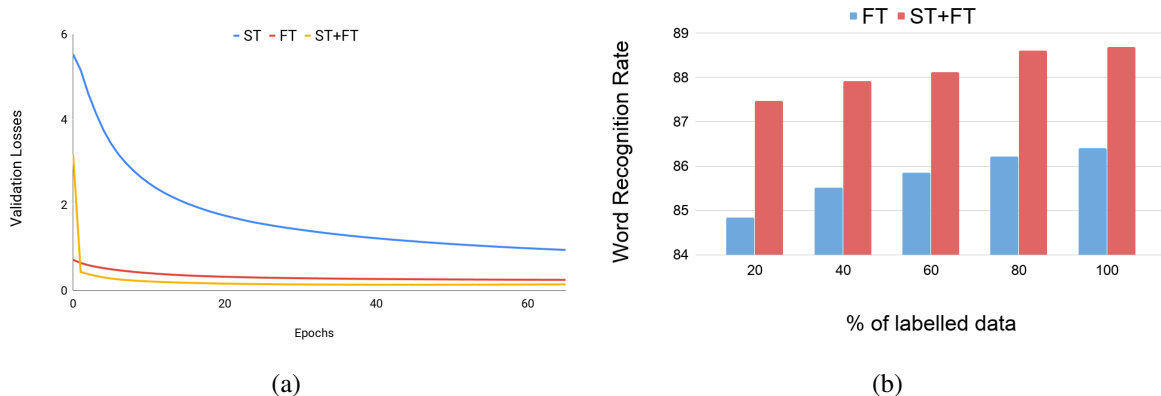


Figure 4.5 (a) Validation losses vs Epochs for self-train, fine-tune and hybrid approaches on English dataset. (b) Comparison of word recognition rates between finetuning and hybrid approach when different percentages of labelled data is used

models are initialized. In case the source and target tasks are very different or else if the source and target data come from very different distributions, then training the model from scratch seems to be a more viable option as compared to finetuning. In order to acclimatize the weights to the source data, we take advantage of the huge quantities of unlabelled data by employing our self-training strategy. We then follow it by finetuning the self-trained models on the limited annotated data. We show by a series of ablations that this approach outperforms the finetuning approach on both the datasets.

4.8.1 Results and Discussions

From Table 4.10 we observe that (1) in the case of Hindi, the improvement in word and character recognition rates are quite significant for our hybrid models with 2% and 1.5% improvement in word accuracy over finetuning and self-training methods respectively. (2) In the case of English, the improvement is significant from self-training. However, the increase in accuracies are not noticeable when compared to the finetuning approach. We attribute it to the fact that the character and word recognition rates had already maxed out using only finetuning, leaving little scope for improvement by the hybrid approach. finetuning on top of self-training has the effect of canceling out any wrong prediction that the model learned during the self-training phase. It also boosts the model’s confidence over uncertain predictions. Hence, we always see a spike in character and word accuracy for the hybrid approach. Figure 4.6a shows the validation curves for the above three approaches. We observe that the validation error rate for the hybrid approach reaches minimum within the stipulated number of epochs and it is closely followed by finetuning. This is in agreement with the results shown in Table 4.10 where the hybrid method outperforms the rest of the two methods. Additionally, we also investigate the effect of the amount of data on finetuning. For this, we systematically increase the amount of labelled data by a factor of 20% to observe the character and word recognition rates for finetuning and hybrid approaches. Figure 4.5b presents the comparison between the hybrid and the finetuning approach for different per-

Method	English		Hindi	
	CRR	WRR	CRR	WRR
finetuning	98.76	98.54	92.87	86.41
Self-train	96.53	94.01	93.04	87.23
Hybrid	98.80	98.64	93.65	88.68

Table 4.10 Character and word recognition rates for finetuning, self-training and hybrid approach on English and Hindi datasets.

Image	Base Model	ST Model	ST+FT
	surcss ✗	stress ✓	stress ✓
	aivailality ✗	availality ✗	availability ✓
	debatec ✗	debate ✓	debate ✓
	windovu ✗	windows ✓	windows ✓
	druwing-roon ✗	drawing-room ✓	drawing-room ✓
	sveetheqrt ✗	sveetheart ✗	sweetheart ✓
	magician ✓	magician ✓	magieian ✗

(a)

Image	Base Model	ST Model	ST+FT
	कयज्यान ✗	बदजबान ✓	बदजबान ✓
	मोमालिब ✗	मनोमलिन्य ✗	मनोमालिन्य ✓
	आप्रापफ ✗	आक्रामक ✓	आक्रामक ✓
	गणणाति ✗	गणपति ✓	गणपति ✓
	ककता-कता ✗	बकता-हकता ✗	बकता-झकता ✓
	उर्वाजिक्र ✗	सर्वसिद्वकर ✗	सर्वसिद्वकर ✗
	मदुछ्य ✗	मनुष्य ✓	मनुष्व ✗

(b)

Figure 4.6 Qualitative Result of our Base, self-trained and hybrid model for English (left) and Hindi (right) datasets. Here ST+FT refers to the model trained using the proposed hybrid approach. We observe that there is a systematic decrease in the character errors from the base model to the hybrid model. This shows that the hybrid model is the best performing model as it has the ability to correct the character errors incurred by the self-trained model due to the confirmation bias. Here crosses show the incorrect words while the correct words are denoted by tick marks.

centages of labelled data. We observe that by (3) finetuning the self-trained model on only a fraction of the labelled data helps us achieve better word recognition rate when compared to the models which were trained using the finetuning approach alone.

4.9 Summary

In this chapter, we presented three different approaches for knowledge transfer between source and target data and compared their performance in terms of character and word recognition rates on two different datasets. We also demonstrated that simple regularization measure on self-training enables the models to learn more efficiently without getting biased towards the noisy network predictions. Additionally, we also showed that combining self-training and finetuning can significantly boost the performance of the OCR across datasets of different languages even when the amount of labelled samples is considerably less.

Chapter 5

Summary and Future Works

In this work, we looked at the problem of boosting the performance of an OCR with minimal human engagement. To this effect, we discussed two approaches. In the first approach, we discussed the problem from the viewpoint of OCR post-processing where we explored the batch correction technique to correct multiple similar error words with minimum human effort. We grouped the error predictions using a novel feature which combined features of word images extracted from a pre-trained CNN and text-predictions outputted by our CNN-RNN based OCR model. We introduced two different correction schemes, automatic and human-aided. We provided experimental results for both the methods and showed that the human-aided approach achieved the best results in terms of correction cost. We, however, show that the proposed automatic correction technique improves with the size of the document collection and it becomes a viable alternative to the human-aided approach if we have a sufficient number of books in our collection. In both the approaches, the correct word was chosen by performing a simple majority voting on the batch of error words. The downside of such a simple technique was that it picked the wrong candidate every time the number of incorrect predictions outnumbered the correct predictions in a batch. In our future works, we would like to explore the role of the OCR confidence score on candidate selection. It would help us discard all such candidates in the batch on which the OCR is not very confident. We believe this will screen out many potential incorrect candidates from the error cluster and help in picking the right candidate for label propagation.

In Chapter 4, we looked at the problem of domain gap between source and target datasets and its effect on the performance of an OCR. To handle such domain shift we made use of two well-known techniques from machine learning literature, finetuning and self-training. We compared the two methods empirically and showed that with right regularization techniques, self-training method can achieve results similar to the finetuning methods with no additional annotation cost. We also showed that by combining self-training with traditional finetuning we can achieve a significant gain in OCR word accuracy compared to the individual finetuning or self-training methods. We further, saw the role of languages in finetuning. We showed that language can act as a reasonable prior in choosing datasets for finetuning. We saw that for some languages there exists a strong correlation amongst them which can help in efficient knowledge transfer from one language dataset to the other language dataset. This

proved to be useful in a setting where a particular language had less annotated data. For our future works, we would like to explore the role of recurrent layers in transfer learning. We feel that not enough work has been done in this domain. Although, we now understand how CNNs respond to finetuning by visualizing their activation maps. The same cannot be said for RRNNs and LSTMs. We hope that by performing an in-depth analysis of how the hidden states of RNNs respond when they adapt from one dataset to other, we will gain a better understanding of transfer learning in sequence to sequence setting.

Related Publications

1. **Deepayan Das** and C.V. Jawahar **Adapting OCR with Limited Supervision.** *Proceedings of International Workshop on Document Analysis System (DAS), 2020*
2. **Deepayan Das**, Jerin Philip, Minesh Mathew and C.V. Jawahar **A Cost Efficient Approach to Correct OCR Errors in Large Document Collections.,** *Proceedings of International Conference on Document Analysis and Recognition (ICDAR), 2019*

Bibliography

- [1] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *PAMI*, 2016.
- [2] Schutze Manning, Raghavan. Introduction to information retrieval. 2009.
- [3] Google Books. Google Books. www.books.google.co.in, 2004.
- [4] Gutenberg. Project Gutenberg. www.gutenberg.org.
- [5] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing using MATLAB*. 2004.
- [6] Wacef Guerfali and Réjean Plamondon. Normalizing and restoring on-line handwriting. *Pattern Recognition*, 1993.
- [7] Partha Pratim Roy, Ayan Kumar Bhunia, Ayan Das, Prasenjit Dey, and Umapada Pal. Hmm-based indic handwritten word recognition using zone segmentation. *Pattern recognition*, 2016.
- [8] Minesh Mathew, Ajeet Kumar Singh, and CV Jawahar. Multilingual ocr for indic scripts. In *DAS*, 2016.
- [9] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- [10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.
- [11] Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, Latesh Malik, Mohantapash Kundu, and Dipak Kumar Basu. Performance comparison of svm and ann for handwritten devnagari character recognition. *arXiv preprint arXiv:1006.5902*, 2010.
- [12] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.
- [13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

- [14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [15] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 1994.
- [16] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *NIPS*, 2009.
- [17] Naveen Sankaran, Aman Neelappa, and CV Jawahar. Devanagari text recognition: A transcription based formulation. In *ICDAR*, 2013.
- [18] Anupama Ray, Sai Rajeswar, and Santanu Chaudhury. Text recognition using deep blstm networks. In *2015 eighth international conference on advances in pattern recognition (ICAPR)*, 2015.
- [19] Karen Kukich. Techniques for automatically correcting words in text. *ACM-CSUR*, 1992.
- [20] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, 1966.
- [21] Hisao Niwa, Kazuhiro Kayashima, and Yasuharu Shimeki. Postprocessing for character recognition using keyword information. In *MVA*, 1992.
- [22] Michael Wick, M Ross, and E Learned-Miller. Context-sensitive error correction: Using topic models to improve ocr. In *ICDAR*, 2007.
- [23] Youssef Bassil and Mohammad Alwani. OCR context-sensitive error correction based on Google Web 1T 5-gram data set. *AJSR*, 2012.
- [24] Sankaran and Jawahar. Error detection in highly inflectional languages. In *ICDAR*, 2013.
- [25] Vinita VS and CV Jawahar. Error detection in indic ocRs. In *DAS*, 2016.
- [26] U Pal, Pulak K Kundu, and Bidyut Baran Chaudhuri. Ocr error correction of an inflectional indian language using morphological parsing. *J. Inf. Sci. Eng.*, 2000.
- [27] B Chaudhuri, U Pal, and P Kundu. Non-word error detection and correction of an inflectional indian language. In *Symposium on Machine Aids for Translation and Communication (SMATAC-96)*, 1996.
- [28] Gregory B Newby and Charles Franks. Distributed proofreading. In *JCDL*, 2003.
- [29] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321, 2008.
- [30] Neeba NV and CV Jawahar. Recognition of books by verification and retraining. In *ICPR*, 2008.

- [31] Ahmad Abdulkader and Mathew R Casey. Low cost correction of OCR errors using learning in a multi-engine environment. In *ICDAR*, 2009.
- [32] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*. Wiley New York, 1973.
- [33] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *ICMLW*, pages 17–36, 2012.
- [34] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*, 2014.
- [35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- [36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [37] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 1965.
- [38] S Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 1967.
- [39] Pramod Sankar K, Vamshi Ambati, Lakshmi Pratha, and CV Jawahar. Digitizing a million books: Challenges for document analysis. In *DAS*, 2006.
- [40] Pingping Xiu and Henry S Baird. Whole-book recognition using mutual-entropy-driven model adaptation. In *DRR*, 2008.
- [41] Youssef Bassil and Mohammad Alwani. OCR Post-Processing Error Correction Algorithm using Google Online Spelling Suggestion. *CoRR*, 2012.
- [42] Rohit Saluja, Devaraj Adiga, Parag Chaudhuri, Ganesh Ramakrishnan, and Mark Carman. Error Detection and Corrections in Indic OCR Using LSTMs. In *ICDAR*, 2017.
- [43] Ray Smith. Limits on the application of frequency-based language models to ocr. In *ICDAR*, 2011.
- [44] K Pramod Sankar and CV Jawahar. Probabilistic reverse annotation for large scale image retrieval. In *CVPR*, 2007.
- [45] Henry S Baird, Venugopal Govindaraju, and Daniel P Lopresti. Document analysis systems for digital libraries: Challenges and opportunities. In *DAS*, 2004.
- [46] Kazem Taghva, Julie Borsack, and Allen Condit. Evaluation of model-based retrieval effectiveness with OCR text. *ACM-TOIS*, 1996.

- [47] Venkat Rasagna, Anand Kumar, CV Jawahar, and Raghavan Manmatha. Robust recognition of documents by fusing results of word clusters. In *ICDAR*, 2009.
- [48] Pramod Sankar K, CV Jawahar, and Raghavan Manmatha. Nearest neighbor based collection OCR. In *DAS*, 2010.
- [49] Praveen Krishnan and CV Jawahar. HWNet v2: An Efficient Word Image Representation for Handwritten Documents. *arXiv preprint arXiv:1802.06194*, 2018.
- [50] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM-STOC*, 1998.
- [51] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. In *PAMI*, 2017.
- [52] Thomas M Breuel, Adnan Ul-Hasan, Mayce Ali Al-Azawi, and Faisal Shafait. High-performance ocr for printed english and fraktur using lstm networks. 2013.
- [53] Naveen Sankaran and CV Jawahar. Recognition of printed devanagari text using blstm neural network. In *ICPR*, 2012.
- [54] Mohit Jain, Minesh Mathew, and CV Jawahar. Unconstrained scene text and video text recognition for arabic script. In *ASAR*, 2017.
- [55] Kartik Dutta, Praveen Krishnan, Minesh Mathew, and CV Jawahar. Offline handwriting recognition on devanagari using a new benchmark dataset. In *DAS*, 2018.
- [56] Wei Liu, Chaofeng Chen, Kwan-Yee K Wong, Zhizhong Su, and Junyu Han. Star-net: a spatial attention residue network for scene text recognition. In *BMVC*, 2016.
- [57] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *CVPR*, 2016.
- [58] Linda Studer, Michele Alberti, Vinaychandran Pondenkandath, Pinar Goktepe, Thomas Kolonko, Andreas Fischer, Marcus Liwicki, and Rolf Ingold. A comprehensive study of imagenet pre-training for historical document image analysis. *ICDAR*, 2019.
- [59] Bruno Stuner, Clément Chatelain, and Thierry Paquet. Self-training of blstm with lexicon verification for handwriting recognition. In *ICDAR*, 2017.
- [60] Volkmar Frinken, Andreas Fischer, Horst Bunke, and Alicia Fournes. Co-training for handwritten word recognition. In *ICDAR*, 2011.
- [61] Christian Reul, Christoph Wick, Uwe Springmann, and Frank Puppe. Transfer learning for ocropus model training on early printed books. *CoRR*, 2017.

- [62] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *ACL*, 2017.
- [63] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.
- [64] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017.
- [65] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *ACL*, 2006.
- [66] Roi Reichart and Ari Rappoport. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL*, 2007.
- [67] Yingting Li, Lu Liu, and Robby T Tan. Certainty-driven consistency loss for semi-supervised learning. *CoRR*, 2019.
- [68] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2017.
- [69] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.
- [70] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *ICLR*, 2017.
- [71] Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves imagenet classification, 2019.
- [72] Eric Arazo, Diego Ortego, Paul Albert, Noel E O’Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. *ICLR*, 2019.
- [73] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *NIPS*, 2019.
- [74] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.
- [75] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *ICLR*, 2017.