OFFLINE HANDWRITING RECOGNITION USING DEEP LEARNING WITH EMPHASIS ON DATA AUGMENTATION EFFECTS

by
FIRAT KIZILIRMAK

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Sciences

Sabancı University
July 2022

FIRAT KIZILIRMAK 2022 ©

All Rights Reserved

ABSTRACT

OFFLINE HANDWRITING RECOGNITION USING DEEP LEARNING WITH EMPHASIS ON DATA AUGMENTATION EFFECTS

FIRAT KIZILIRMAK

Computer Science and Engineering MS.c. THESIS, July 2022

Thesis Supervisor: Prof. AYŞE BERRİN YANIKOĞLU

Keywords: offline, handwriting recognition, deep learning, data augmentation

We have proposed a deep learning model leveraging train and test time data augmentation approaches for the problem of offline handwriting recognition. We made a comprehensive analysis using our CNN-BiLSTM network to provide evaluation results of each component of the network for the IAM dataset, which is the most commonly used handwriting dataset. We experimented with the deep learning network architecture; evaluated the effects of train time data augmentation and pretraining the network with synthetic handwriting images to alleviate the data sparseness problem. We proposed two different test time augmentation methods as post-processing approaches to obtain the correct transcription of the handwriting image out of the partly correct predicted transcriptions. While the test time augmentation increases the time and computational complexity, it reduced the error rate by 2.5% points and thus could be preferred for batch processes when there are no real-time recognition requirements.

Furthermore, we attempt the initial steps of offline handwriting recognition for the Turkish language. To this end, we crafted the first, line-level Turkish handwriting dataset, consisting of more than 2600 line images collected from 73 different people. We applied our deep learning method to this dataset to provide baseline results for future studies; besides, we explored approaches including transfer learning from the IAM to the Turkish dataset and joint training with both datasets.

Our contributions include an extensive error analysis for both datasets, revealing better insights into methods and datasets. As part of this effort, we provide our evaluation criteria clearly and completely along with our proposed model to encourage scientific reproducibility.

ÖZET

VERI ARTIRMA ETKILERINI VURGULAYARAK DERIN ÖĞRENME YÖNTEMLERI ILE GÖRÜNTÜDEN EL YAZISI TANIMA

FIRAT KIZILIRMAK

BİLGİSAYAR BİLİMİ VE MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, TEMMUZ 2022

Tez Danışmanı: Prof. Dr. AYŞE BERRİN YANIKOĞLU

Anahtar Kelimeler: çevrimdışı, el yazısı tanıma, derin öğrenme, veri artırma

Biz bu çalışmada çevrimdışı el yazısı tanıma problemi için eğitim ve test sırasında veri artırma tekniklerinden yararlanan bir derin öğrenme modeli önerdik. Bu alandaki en sık kullanılan el yazısı veri kümesi olan IAM veri kümesi üzerinde derin öğrenme ağının her bir bileşenini değerlendirmek için CNN-BiLSTM ağımızı kullanarak kapsamlı bir analiz yaptık. Derin öğrenme ağı mimarisi ile deneyler yaptık; veri kıtlığı problemini hafifletmek için eğitim zamanı veri artırma ve ağı sentetik veriler ile ön eğitimden geçirmenin etkilerini değerlendirdik. El yazısı görüntüsüne ait eksik ya da hatalı tahmin edilen çıktılardan doğru çıktıyı elde etmek için bir son işleme yöntemi olarak iki farklı test-zamanı veri artırma metotu önerdik. Bu test zamanı veri artırma yöntemi sistemin zaman ve çalışma karmaşıklığını artırsa da hata oranlarını %2.5 oranında düşürmüştür ve bu sebeple gerçek zamanlı tanıma gereksinimlerinin olmadığı, veriyi yığın olarak işleme durumlarında kullanılabilir.

Ayrıca, Türkçe dili için çevrimdışı el yazısı tanımanın ilk adımlarını atıyoruz. Bu amaçla 73 farklı kişiden toplanan 2600'den fazla satır görüntüsünden oluşan ilk, satır düzeyinde Türkçe el yazısı veri kümesini oluşturduk. Geliştirdiğimiz derin öğrenme yöntemini bu küme üzerinde de uygulayarak gelecek çalışmaların temel alabilecekleri sonuçlar sağladık; ayrıca, IAM kümesinden bizim topladığımız Türkçe el yazısı veri kümesine öğrenme aktarımı ve her iki veri kümesi ile ortak eğitim gibi yaklaşımları araştırdık.

Katkılarımız her iki küme için de yöntemlere ve kümelere dair daha iyi içgörüler sunan itinalı bir hata analizini de kapsar. Bu çabanın bir parçası olarak da bilimsel tekrarlanabilirliği teşvik etmek için önerilen modelimiz ile birlikte değerlendirme kriterlerimizi açık ve eksiksiz bir şekilde sunuyoruz.

ACKNOWLEDGEMENTS

I would like to thank the people who helped and supported me throughout my thesis journey for the last two years. First and foremost, I want to express my sincere gratitude to my supervisor Prof. Ayşe Berrin Yanıkoğlu for her guidance, support, and feedback. Throughout this thesis, I learned a lot from her immense knowledge and experience on the topic, besides other academic and personal development contributions I received.

Also, I would like to thank my friends for supporting and encouraging me whenever I stumble. I am indebted to them and my brothers for their help during the collection and preparation of the Turkish handwriting dataset as well as for their proofreading and feedback.

Lastly but most importantly, I would like to express my deepest gratitude to my family. I am fortunate to be raised in such a family who have supported and loved me unconditionally. They always listened, encouraged, and helped me, whenever I needed; they deserve my endless gratitude.

This thesis work is dedicated to my beloved family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

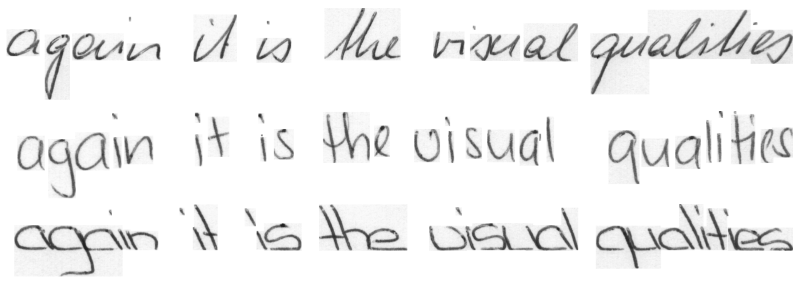
1 INTRODUCTION

Offline handwriting recognition refers to the problem of recognizing a handwritten text in an image. This is a subfield of optical character recognition (OCR) in which the text to be recognized is handwritten rather than printed or irregular scene text. Thus, this domain has specific challenges while sharing common characteristics with OCR, and thereby, these challenges make offline handwriting recognition a long standing and still an active research field.

Initial attempts at this problem mainly used Hidden Markov Models (HMMs) with hand-crafted features while making use of language processing techniques [9, 43]. However, deep learning models have become the method of choice for the handwriting text recognition (HTR) problem, in the last decade [13, 20, 35, 39, 49]. In particular for the offline handwriting recognition, image based character sequence generation methods have been studied. Usual steps of these methods involve processing input handwriting images, producing sequence of features maps, and then generating sequence of characters, which is expected to be corresponding transcription of the input handwriting.

Most of the works, in the last decade, have used Convolutional Neural Networks (CNNs) in combination with recurrent neural networks (RNNs) and optimized the entire architecture using Connectionist Temporal Classification (CTC) loss function [19]. This approach allows the model to be end-to-end trainable without requiring explicit image-character alignment, which makes it favorable over traditional models. Moreover, the CTC algorithm makes many-to-one alignment in which more than one input time frame could map to a single time frame at the output, allowing characters that do not fit in a single time step to be recognized. This property of the CTC function enforces the output time frame length to be less than or equal to of input time frame. Therefore, others have modelled the problem as sequence-to-sequence and proposed attention based encoder-decoder models [30, 32, 39] due to the success of attention based approaches in sequence related areas [4, 55].

The first sequence-to-sequence methods have combined a CNN with a bi-directional RNN, as the encoder, and a one-directional RNN with an attention mechanism, as the decoder. More recent approaches utilized Transformer decoders [13] for decoding or Vision Transformers [35] as a sequence-to-sequence architecture to exploit their applicability over handwriting recognition. The attention mechanism leverages the relationship between the characters in a line and could provide features encoded with language-based knowledge.



again it is the visual qualities
again it is the visual qualities
again it is the visual qualities

Figure 1.1 The same text written by different people, showing the possible dissimilarities of handwriting. Examples from IAM dataset [37].

Issues in Offline Handwriting Recognition

Despite all the state of the art works, the handwriting recognition field still lacks from sufficient amount of data containing handwriting images in various styles. Moreover, handwriting recognition has its own difficulties. For instance, writing style (e.g. cursive or slanted) varies across different people and it is even possible to find dissimilarities in a writing of the same person. Figure 1.1 displays an example variation of handwriting styles for the same text written by three different people. This leads us to a data sparseness problem in terms of capturing distinct writing styles. In addition, most of the existing datasets comprise historical handwriting images, making it impractical to train and evaluate a model for modern handwriting cases.

There are two common approaches in order to alleviate the data sparseness: augmenting images at train time and generating synthetic handwriting images to pretrain the models [15, 35, 36, 39, 58]. Handwriting images are transformed to imitate the same text as if it is written in a different style by spoiling letter shapes while preserving the readability. This way, it becomes possible to increase the diversity of handwriting styles. On the other hand, synthetic images play an important role in offline handwriting recognition as they introduce artificially written handwriting styles to increase variability further. Due to these advantages, both strategies are widely used for introducing different writing styles to increase the generalization capacity of models.

Furthermore, the majority of the works in this field study handwriting recognition for the English language. Therefore, there is a limited number of studies for other languages including Turkish. In addition to being a low-resourced language, the Turkish language has a complex, agglutinative morphological structure. That is, there is a practically infinite number of words that could be generated by appending morphemes in sequence, which limits the success of lexicon-based approaches. Hence, offline Turkish handwriting recognition has not been explored adequately except in the very first attempts.

Our Motivation and Contributions

This thesis contributes to the offline handwriting recognition problem in the following aspects:

i) We make a comprehensive analysis using deep learning models on the line-level IAM dataset [37]:

   - Experimented with deep learning networks with different parameters and architecture design in order to illustrate their effect on recognition performance.

   - Evaluated the effectiveness of train time data augmentation methods along with synthetic data generation. Following the studies in this domain, we applied similar transformations with combined and in separate ways to measure their importance properly.

   - Generated almost 2.5 million synthetic handwriting lines in the English language in order to reduce the data sparseness and demonstrate their effectiveness.

ii) We propose a simple yet effective test time augmentation technique to obtain better performance that can be preferred when the runtime is not an issue (as in batch/offline applications):

   - Developed and assessed two methods, underlined the one with simple yet effective approach, and pointed to potential improvements that could be possible using a better function.

   - Measured running times of the proposed method on two different machines to show the trade-off between running time and accomplished success.

iii) We introduce the first, line-based Turkish handwriting dataset consisting of 2600 line images written by 73 different people.

   - Applied our method on this dataset to provide a benchmarking baseline for future works.

iv) We provide a careful error analysis of our methods on IAM and Turkish datasets and inspect dataset related issues.

v) We share our code baseline[1] as well as our line-level Turkish handwriting dataset to encourage scientific reproducibility.

[1]https://github.com/firatkizilirmakk/handwriting-recognition

Thesis Structure

The thesis is organized as follows:

Chapter 2 describes the related work in the offline handwriting recognition domain, from recognition models to data augmentation and offline Turkish handwriting recognition studies.

Chapter 3 elaborates the methodology followed through the thesis. The chapter gives detail on both the deep learning model and data relevant topics including train & test time augmentation, synthetic data generation, and Turkish handwriting dataset collected for this work.

Chapter 4 contains the conducted experiments throughout this work and the corresponding results. The chapter also includes an error analysis on the IAM dataset.

Chapter 5 concludes the thesis with the summary of the work and points to future research directions.

2 RELATED WORK

The offline handwriting recognition problem has been studied for decades. Initial attempts combined Hidden Markov Models with hand-crafted image-based features for recognizing handwritten text [9, 43, 44]. Later, successor applications tried Hidden Markov Models in conjunction with Artificial Neural Networks (ANNs) to make use of their strengths together.

However, starting with the presentation of the Connection Temporal Classification algorithm [19], research in this domain has gained increasing attention. Afterwards, deep learning methods have been proposed and achieved greater success than ever before, and still, errors get reduced with the advancements of the deep learning models [13, 15, 30, 35, 39, 49] and data related techniques [36, 45, 58, 59]. Below, we elaborate on related works in the offline handwriting recognition domain in terms of models, data as well as Turkish handwriting studies.

Offline Handwriting Recognition Models

Hidden Markov Models

The fundamental approach to handwriting recognition prior to the deep learning era was with Hidden Markov Models [44]. HMMs are doubly statistical models for the observable output and the underlying hidden states. The fundamental idea is to generate a sequence of characters or words, $\mathbf{S}$, from a sequence of feature vectors, $\mathbf{X}$, that is encoded with visual features of input handwriting images. Formally the aim is to maximize the posterior probability $P(\mathbf{S} \mid \mathbf{X})$, which is formalized within Bayes' Rule [5].

Preliminary studies explored offline handwriting recognition with a sliding window approach for feature extraction and Hidden Markov Models for generating character or word sequences [9, 43]. They further supported their models with external language models and investigated the effects of lexicon during decoding.

The HMM approach has been utilized in other sequence-related problems such as automatic speech recognition (ASR) [5, 34] and online handwriting recognition [25]. However, the Hidden Markov Models and the corresponding literature is beyond the scope of this thesis.

CTC Based Models

Later, Connectionist Temporal Classification (CTC) method was introduced for speech recognition task [19], allowing ANN models to be trained end-to-end with the backpropagation [57] algorithm for sequence classification without any need of pre-segmented data. The method was adopted for handwriting recognition [20] where they (1) extract sequence of image features using sliding window with 1 pixel width. (2) feed these features to a bi-directional LSTM (BiLSTM) [24]. (3) produce character sequences via CTC layer. They significantly outperformed HMM and HMM-RNN based approaches with the proposed BiLSTM + CTC model. Further, Graves et al. applied multi-dimensional LSTM (MDLSTM) layers instead of BiLSTM to incorporate more context around letters and to obtain better transcriptions [21].

With the rise of deep learning and after great success of CNN models on image processing tasks [23, 33], deep learning methods have been explored for handwriting recognition problem as well. Instead of using hand-crafted image features [20, 21], Shi et al integrated a CNN architecture to produce more robust image features [49]. The method first processes input handwriting image with the CNN, generates sequence of image features, and feeds them through BiLSTM + CTC layers to obtain final transcription. Inspiring from these, Bluche et al proposed a gated convolutional model for computing more generic image features [8]. Puigcerver [46] showed the effectiveness of single dimensional LSTM layers over multi dimensional ones. Further, Dutta et. al. [15] made a comprehensive study demonstrating effects of data augmentations, pretraining and use of Spatial Transformer Network (STN) [27].

Sequence-to-Sequence Approaches

The above methods employed CTC layer to obtain transcription of the input handwriting from model outputs. However, CTC method has a drawback that disallows to generate longer sequences than input sequence, which limits the CNN architecture to be used [30, 39]. As feature maps get smaller, due to convolution and max pooling operations, the generated sequence becomes shorter, which in turn could result in missing transcriptions. Hence, attention based sequence-to-sequence methods have been developed in order to overcome the shortcomings of the CTC and to leverage their sequence learning capabilities on handwriting recognition [30, 31, 32, 39]. The fundamental idea with these methods is to use CNN + RNN, usually a BiLSTM, to encode input handwriting as a sequence of features, and then employ attention based RNN, usually an LSTM or a GRU, to decode the encoded sequence. The overall architecture is optimized with the cross entropy loss function, applied over each frame through the output sequence.

Micheal et al [39] utilized CNN + BiLSTM to encode input handwriting image and attention based LSTM to decode the encoded representation into sequence of characters. They compared different attention mechanisms such as content-based, location-based and penalized attention, and further combined the CTC loss along with the cross entropy loss to increase model capabilities. Kass et al [32] integrated the Spatial Transformer Network at the forefront of their sequence-to-sequence model to reduce handwriting variations before feeding images to the rest of the architecture. Apart from these, Kang et al [30] incorporated a character level language model into training phase where they feed the attention based LSTM decoder with the concatenation of encoder and language model outputs.

More recently, Transformer [55] based models have been presented for handwriting recognition due to their substantial achievements on sequence related tasks [13, 35]. Diaz et al compared CTC and sequence-to-sequence approaches, experimented with Transformer decoder and found the best performing model with self-attention encoder + CTC. Li et al, on the other hand, employed Vision Transformer [14] as an encoder and vanilla Transformer decoder, taking advantage of pretrained Vision Transformer and Transformer decoder. They obtained the state of the art results and further showed the effectiveness of their model and pretraining scheme, without any post processing or external language model.

Data Augmentation & Synthetic Data Generation

Lack of high quality, modern handwritten text data limits the capabilities of these advanced models and thereby disallows generalisable offline handwriting recognition. Therefore, most of the studies in this domain proposed solutions for dealing with data sparseness of handwriting recognition. There are two common approaches: (1) applying data augmentation techniques at train time for introducing different handwriting styles. (2) generating synthetic handwriting images to provide enough and diverse handwriting styles to deep learning models.

Affine transformations such as rotating, scaling, and shearing are heavily applied and are shown to be effective methods for mimicking handwriting styles [15, 45, 59]. Apart from the affine transformations, Wigington et al [58] developed a distortion method along with a profile normalization technique for spoiling parts of letters to be more discriminative. Further, Luo et al [36] proposed to learn an augmentation as a joint-task during training of the models and further provided their closed-form transformations to incorporate into training of models.

On the other hand, generating synthetic handwriting images plays a crucial role due to lack of open access, high quality, and modern handwriting image datasets. All the studies have generated their own synthetic data, either in word or line level, and experimentally shown their effectiveness. While some [15, 30, 59] have synthesized a few millions of handwriting images using words or sentences from widely used large corpora including Brown and Lob corpora [1, 17]; others used synthetic data generated for scene-text recognition problem [32], or cropped lines from pdf files containing handwritten text [35]. However, though the above methods gained performance increase using synthetically generated data, none of them published their dataset to alleviate the data scarceness problem and ease reproducibility.

Test Time Augmentation

In addition to augmenting images at train time, there is a counterpart approach of it called test time augmentation. The idea is to transform an image at test phase so as to obtain possibly different outcomes and then decide on the best one. It is widely used for image classification task where an input image is augmented during testing and the results are combined for a better prediction [48].

In the case of offline handwriting recognition, there are a few studies involving test time augmentation method in which they followed similar approaches to each other [15, 45, 58]. Poznanski et al [45] applied 36 different transformations on an image at test time, retrieved model outputs for these augmented images and the original one, and afterwards took the mean of these outputs as the final outcome. While Dutta et al [15] followed the same approach in [45], Wigington et al [58] employed 20 transformations, obtained the corresponding transcriptions and picked the final one with respect to the lowest CTC loss value.

Offline Turkish Handwriting Recognition

Offline handwriting recognition has been mostly studied for the English language. Therefore, in this domain, languages like Turkish suffer from lack of high quality data, applications of advanced models and benchmarking baselines for future studies to compare with.

Rather than offline handwriting recognition, previous works mainly focused on online handwriting recognition [7, 56]. The online recognition requires data in the form of time series signals perceived by pen-based tablet or mobile phones. One of the very first attempts were to obtain features by hand for each point of the time series signal and employ a Hidden Markov Model to recognize a written word [56]. Later, another HMM based model was proposed [7] where they concentrated on data sparseness of the Turkish language and proposed a preprocessing method along with a recognition of sublexical words instead of words themselves.

For the offline Turkish handwriting recognition problem, Yanikoglu et al [60] introduced the first method applying a Hidden Markov Model with a prefix recognizer to deal with Turkish handwritten character and word recognition. Another Turkish character recognizer was proposed in [10] extracting image base features by hand and then classifying these features as Turkish characters using methods including Artificial Neural Networks and K-Nearest Neighbor. Aydemir et al [2] developed Arabic-Turkish handwritten word recognizer.

To the best of our knowledge, there is only one work that used deep learning methods to recognize Turkish handwritten characters, as introduced in [6]. Therefore, apart from the other contributions, we believe that this thesis work is the first to leverage advanced deep learning models on offline Turkish line-based handwriting recognition problem as well as the first attempt to collect and share line-based Turkish handwriting dataset.

3 METHODOLOGY

In this section, we explore our approach to the offline handwriting recognition problem. We intended to use an accomplished deep learning network for this problem while exploiting the submodules constituting the model. While doing so, we utilized subsidiary methods including data augmentation and synthetic data generation to reduce the error rates of our model. On top of these, we propose two separate test time data augmentation methods to further lower the errors without adding any model-based complexity or requirement for more data. Lastly, we introduce an offline Turkish line-based handwriting dataset, which we collected for this thesis, and further describe our baseline method on this dataset for upcoming studies to benchmark with.

3.1 Deep Learning Model

Inspired from [49], and its later accomplished successors [8, 15, 46], we followed a similar deep learning architecture. The architecture consists of three fundamental blocks: feature extraction from the input image using convolutional layers; processing the extracted deep features as a sequence using bidirectional LSTMs; and producing the sequence of output characters with the help of CTC decoding. These modules are described in detail below and the network structure is depicted in Figure 3.1.

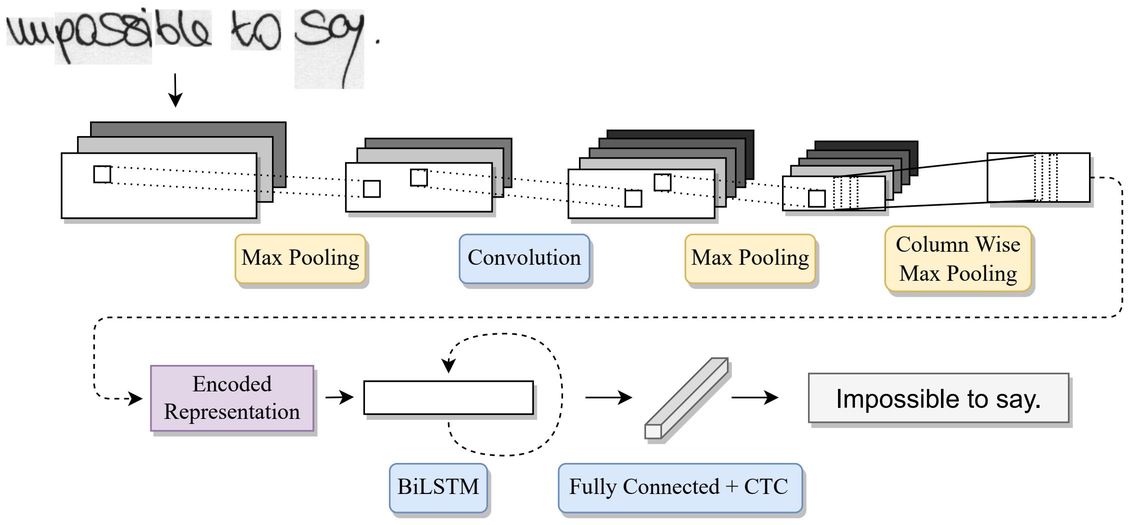


Figure 3.1 Our proposed deep learning network consisting of CNN-BiLSTM models as encoder and CTC as decoder.

3.1.1 Feature Extraction

The network uses 12 convolutional layers with 3×3 kernels for feature extraction. Max pooling operation, with 2×2 kernel, is applied two times in total; after the first two and following four convolution layer blocks. The convolution layers in these blocks have 32, 64 and 128 number of filters. ReLU [41] activation function and batch normalization [26] are applied after each convolution for faster convergence and easier training. The CNN network produces a feature map tensor in the form of $D \times H \times W$ where D corresponds to number of output filters or depth of feature maps; H and W stand for height and width of feature maps, respectively. Then, we apply a max-pooling function over the height dimension and interchange the dimensions, resulting a tensor of $W \times D$.

This output represents D dimensional feature vectors for a sequence of length W. More formally, W could be written as $W = [w_1, w_2, ..., w_t]$, indicating a sequence of t time frames where $w_i \in \mathbb{R}^D$ for $i \in \{1, 2, \ldots, t\}$.

We intentionally avoided using deep CNN networks in order not to narrow down the feature maps too much. Otherwise, the CTC algorithm would only allow producing shorter sequences due to its constraints, as explained in Section 2. Nonetheless, we conducted experiments with a different number of convolution and max-pooling layers and even tried well-known image feature extractors like ResNet18 [23] architecture to decide the baseline model. We explain the outcomes of these experiments in Section 4.1.1.

3.1.2 Sequence Learning

This phase processes the sequence of features crafted by the feature extraction step. The idea is to incorporate sequence learning capabilities of Recurrent Neural Networks to obtain sequences of features representing the input handwritten text better.

Following [8, 15, 46, 49], we utilised a bi-directional LSTM model as well, leveraging learned contexts from both directions. We used two BiLSTM layers having 256 hidden dimensions, without any dropout [52] applied. We feed the BiLSTM layers with the delivered sequence of features, $W \times D$, and the BiLSTM produces a new tensor of size $W \times (2 \times K)$, where K stands for the hidden dimension size of the BiLSTM layer which is equal to 256. The produced tensor, sequence of feature vectors, holds an encoded representation of the input handwriting image in a feature space.

Additionally, we experimented with bi-directional Gated Recurrent Unit (GRU) [11] based models and conducted experiments with different number of recurrent layers and hidden dimension sizes. Section 4.1.2 illustrates the effect of these experiments.

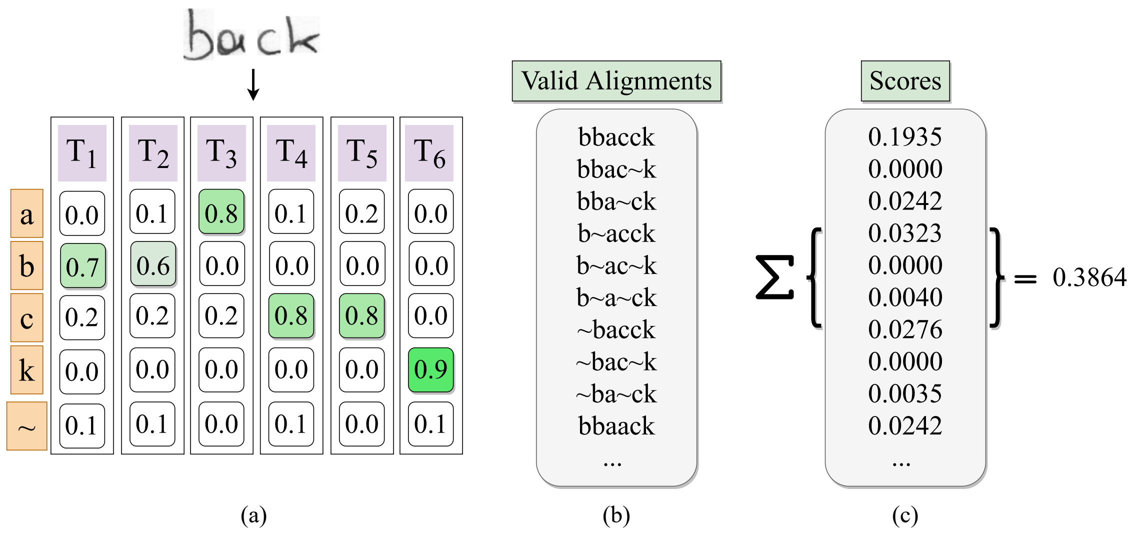


Figure 3.2 Simplified illustration of the CTC algorithm [19] with a toy alphabet consisting of four letters (a, b, c, and k). Symbol ~ stands for the blank token of the CTC method. (a) Probability distributions produced by our deep learning model for the input handwritten image **"back"**, over the 4-character alphabet through six time frames $T_1, T_2, \ldots, T_6$. (b) Sample of possible valid alignments to obtain text **"back"**, computed by the CTC algorithm. (c) Summed probability values of the corresponding alignments. The method then marginalize all these probabilities to generate cost value, which is computed by taking negative logarithm of the summed probabilities. That is equal to $-\log(0.3864) = 0.4129$.

3.1.3 CTC Decoding

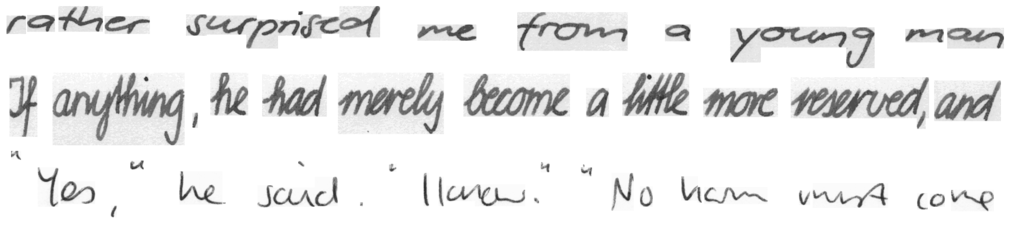
After passing the input image through the first two phases, feature extraction and sequence learning, the encoded representation of the input is obtained in the following form $W \times (2 \times K)$; formally $W = [w_1, w_2, ..., w_t]$ where $w_i \in \mathbb{R}^H$ for $i \in \{1, 2, \ldots, t\}$. Here, H stands for $2 \times K$ or a real value 512 in our case of K being set to 256. Then, the encoded sequence representation, $W \times (2 \times K)$, is mapped to a sequence of probability distributions over alphabets of recognizable characters, resulting in $W \times C$. To do so, we utilised two fully connected layers; the first one is followed by a ReLU activation and the second one outputs through C number of cells. Further, the SoftMax function is applied to produce a sequence of probability distributions defined over the alphabet.

Once the sequence of probability distributions is obtained, it is time for the CTC function to align input and output (ground truth) sequences while computing the cost of the alignment. To this end, CTC function applies the following steps: (1) find valid alignments for the output sequence. (2) sum up probabilities of these valid alignments. (3) return minus logarithm of the summed up probability as the cost of the alignment. The simplified steps of the CTC method, as described above, is depicted in Figure 3.2.

As seen through the summarized steps of the CTC algorithm, there is no need for predefined alignment between input and output sequences. Thus, deep learning networks utilizing the CTC loss function, which is defined as differentiable, could be trained end-to-end using the backpropagation algorithm [57].

3.2 Datasets

Data is another most important component of an offline handwriting recognition system, as in other statistical problems. However, there is almost no development on the data side for a long time despite the model side advancements in this domain. The majority of the works have used IAM [37] and Rimes [22] datasets, containing a word or a line level handwritten text in English and French languages, respectively. Apart from these, there are a few historical datasets, Bentham [53] and Saint Gall [16], that consist of handwritten text written in English and Latin languages through the 18th and the 9th centuries respectively. Figure 3.3 displays example lines from these datasets.

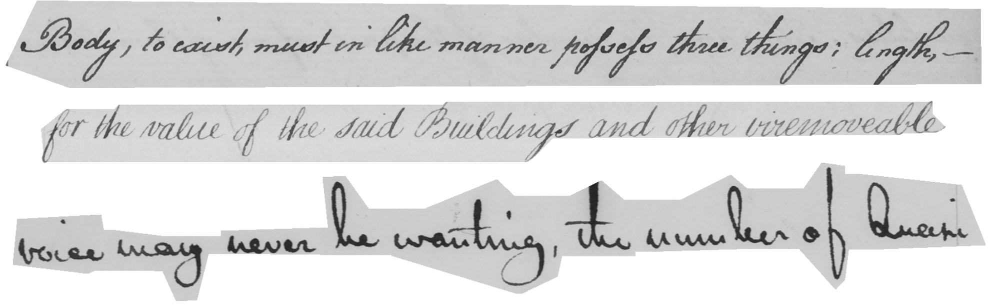


rather surprised me from a young man

If anything, he had merely become a little more reserved, and

"Yes," he said. "I have." "No harm must come

(a) IAM

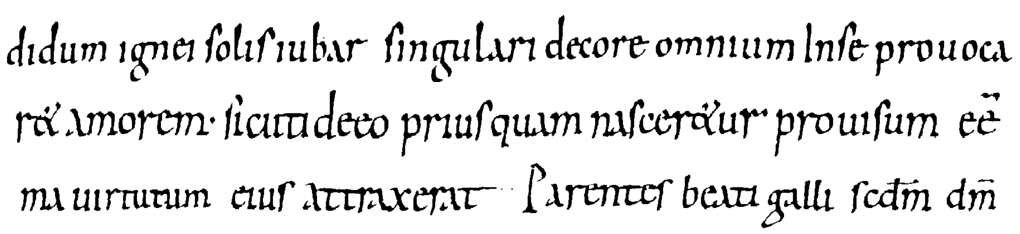


Body, to exist, must in like manner possess three things; length, —

for the value of the said Buildings and other irremoveable

voice may never be wanting, the number of Queri

(b) Bentham



didum ignei ſoliſ iubar ſingulari decore omnium Inſe prouoca

ret amorem· ſicut a deo priuſquam naſceretur prouiſum ee

in a uirtutum eiuſ attraxerat· Parenteſ beati galli ſcdm dm

(c) Saint Gall

Figure 3.3 Sample line images from three public handwriting datasets: IAM [37], Bentham [53], and Saint Gall [16].

IAM dataset was published in 1999 [37], and still is the most commonly used English handwriting dataset. It contains handwriting texts in English written by 657 different people. There are 1,539 scanned pages, partitioned into 10,373 labelled text lines with 79 different characters. We preferred the most commonly used Aachen partition of the dataset, in which the train, validation, and test splits are as follows: 6161 lines for training, 966 for validation, and 2915 for testing. To the best of our knowledge, there is no other public dataset in English with non-historical and line-level handwriting images. Sample line images from the IAM dataset is displayed in Figure 3.3 (a).

Rimes dataset [22] consists of French handwriting images and has been used for evaluating the generalization abilities of models for a long time. However, we have not been able to access the dataset using official channels and the website of the publisher, as also indicated here [1]. Therefore, our experiments do not include results on this dataset.

On the other hand, it is seen in Figure 3.3 (b) and (c) that historical handwriting are not suitable to adopt for modern writings, as in the IAM dataset, Figure 3.3 (a). Therefore, we narrowed down our dataset selection to only using the IAM dataset for the English language and also to our Turkish line-based handwriting dataset for the Turkish language, which is elaborated in Section 3.5.

3.3 Data Augmentation & Synthetic Data Generation

Usually, the more data are introduced to a model better the performance is obtained. Even in data abundance, it is suggested to apply data augmentation methods to further increase variability. Hence, data augmentation and synthetic data generation approaches have been studied extensively and shown their effectiveness on most of the tasks [50].

The handwriting recognition field still lacks high-quality, open-access datasets with a high number of samples, even though the problem has been studied for decades. Due to this data sparseness, current methods usually exploit data augmentation and synthetic data generation techniques to make use of their effectiveness. The fundamental idea is to alleviate the effect of data lack and present various handwriting styles for better performance. Yet, these works have not published the synthetic datasets they have crafted, which in turn disallows a common evaluation protocol.

[1] https://github.com/jpuigcerver/Laia/issues/44

3.3.1 Data Augmentation

Although data augmentation for visual tasks has been explored extensively [50], there are a couple of studies that proposed novel approaches to augment handwriting images [36, 51, 58], while others utilized common methods such as affine transformations [15, 45, 58, 59].

Affine transformations are among the most popular data augmentations for handwriting recognition due to their simple yet effective nature [15, 45, 59]. Specifically, rotation, shear, scale, and translation have been applied in almost all the works. Some also have used blurring and sharpening methods [29, 30] along with the elastic distortion approach, proposed in [58] and geometric transformations of [36], to deform handwriting letters for increasing variety of styles. Following these studies, we employed affine conversions including shear and rotation as well as elastic distortion and geometric transformations. Augmented examples with these conversions are displayed in Figure 3.4.

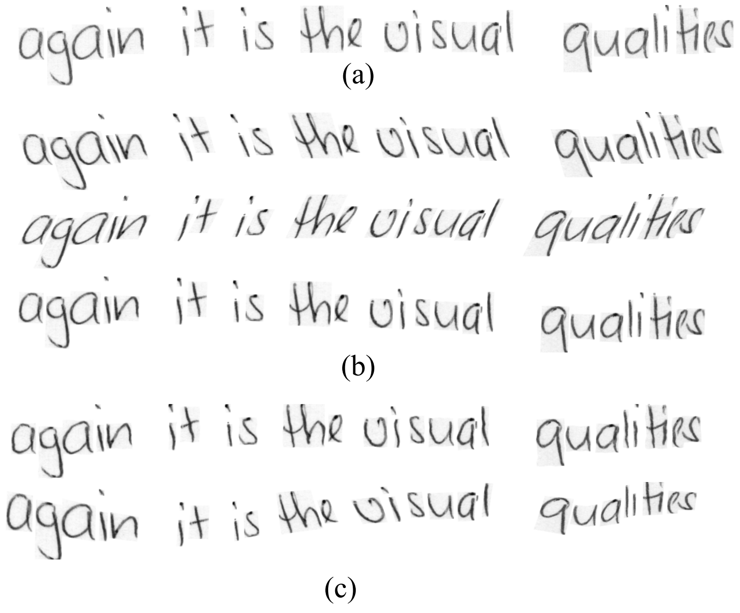

again it is the visual qualities
(a)
again it is the visual qualities
again it is the visual qualities
again it is the visual qualities
(b)
again it is the visual qualities
again it is the visual qualities
(c)

Figure 3.4 Original image along with the augmented samples. (a) Original image. (b) Augmented images by affine transform: the first two lines are sheared, and the last one is rotated. (c) Images with the elastic distortion and the perspective transform from the geometric conversions.

Shear is a linear affine transformation that takes a point $(x,y) \in \mathbb{R}^2$ and maps it to $(x+ky,y) \in \mathbb{R}^2$, if it conducts horizontal conversion. Here k denotes the scaling factor determining how many pixels to move in an image. Rotation is another linear affine transformation that rotates a point (x,y) in the counter-clockwise direction by θ degree. Formally, the rotation is composed of a transformation matrix M:

$$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cost\theta \end{bmatrix}$$

and rotates (x,y) points by θ degree into (x',y') points with:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cost\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Using these shear and rotation transformations makes it possible to mimic slanted handwriting styles as illustrated in Figure 3.4 (b).

Elastic distortion, on the other hand, is a non-linear transformation that elastically changes the shape of a letter. We used the method in [51] where displacement fields Δx and Δy are sampled from a probability distribution. First, they sample x and y positions of the field from a uniform distribution between 1, +1; then, convolve these fields separately with a 1-dimensional Gaussian kernel of σ standard deviation; finally, the fields get multiplicated by α value to determine the distortion intensity. Overall, the method is able to distort handwriting letters using σ and α parameters.

Moreover, geometric transformations, proposed in [36], transform an image by moving a set of predefined points to new positions. The movement of the points is determined via an agent network, where this transformation method is optimized jointly with a text recognition module to craft readable yet hard samples. We used closed-form versions of their augmentations (publicly shared in[2]) namely distortion, stretch, and perspective, and grouped them under the name of geometric transformations.

These transformations were applied more than once with different parameters to generate distinct handwriting styles. We decided to the range of parameters of the augmentation methods by visual inspection,. We deformed the letter shapes to some degree while considering the readability of the handwriting. During training, input images undergo a transformation with a probability p = 0.5, which makes it practically impossible to introduce the same image again.

We trained our base model with and without transforming images in order to evaluate the effect of these augmentation techniques. Outcomes of these experiments are given in detail in Section 4.2.

[2]https://github.com/Canjie-Luo/Text-Image-Augmentation

3.3.2 Synthetic Data Generation

Generating synthetic data plays an important role in a very wide range of tasks from data privacy to unbalanced datasets. The idea is to imitate the original data by approximating the underlying distribution of the data, which, in turn, could mitigate the need for original data. In the case of offline handwriting recognition, synthetic data generation has become a key ingredient of the current methods due to the scarcity problem, as explained above.

Most of the current approaches craft their synthetic dataset following similar steps such as employing texts written in widely used large corpora [1, 17] or crawling text from the internet [30] to synthesize artificially written handwritten images. However, they have not made their dataset public, even though they have demonstrated the effectiveness of using synthetic data, which prevents proper comparisons of methods and scientific reproducibility.

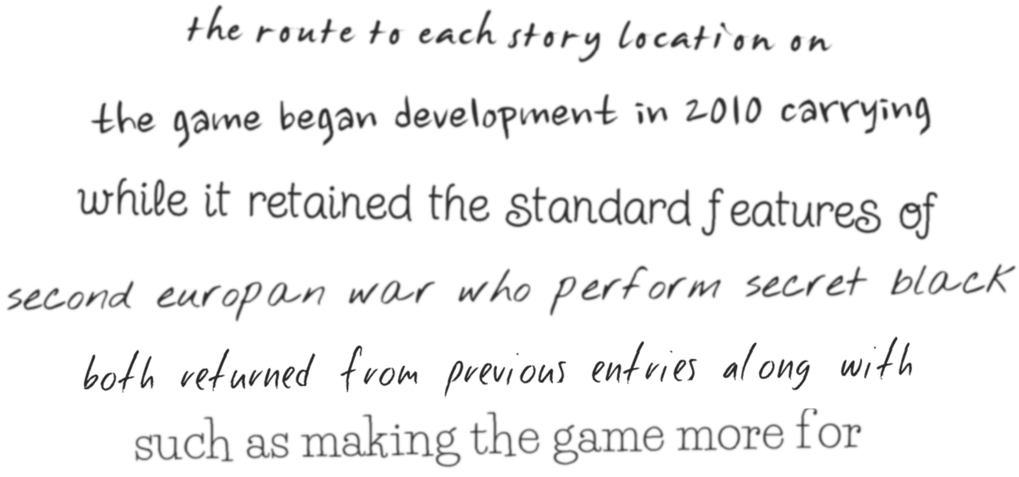

the route to each story location on
the game began development in 2010 carrying
while it retained the standard features of
second europan war who perform secret black
both returned from previous entries along with
such as making the game more for

Figure 3.5 Examples of synthetically generated handwriting line images.

In this work, we have generated our own handwriting dataset as well. We combined WikiText-2 [38], Brown [17], and training text of IAM dataset (which corresponds to LOB corpus [1]) as the corpus to generate images from. Then the texts were preprocessed to have a unified form with correctly tokenized words. The sentences were split to have at least one word and at most 8 words in a line to be generated synthetically. More than 200 handwriting TrueType fonts were collected from the internet and then utilized to synthesize images. We took advantage of the trdg tool [3] to generate synthetic images using these corpora and fonts. Overall, we have generated almost 2.5 million synthetic handwriting image lines, example images are illustrated in Figure 3.5. In Section 4.3, we show empirically the effectiveness of using pretraining data.

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

3.4 Test Time Augmentation

Deep learning networks could make incorrect predictions depending on several reasons. Therefore, inspecting different representations of the same data might reveal a better output. For offline handwriting recognition, the idea of the test time augmentation is to apply transformations to an input handwriting image at the test phase, obtain transcriptions of the corresponding augmented images, and then decide on the best decoding. In this way, we could find a way to produce a better transcription than of the case when only the original image is presented.

Even though test time augmentation is a simple and effective idea, it has not been explored well due to time complexity it adds to the overall decoding process. However, the method could be preferable in the case of offline handwriting recognition, where consuming a bit more time could be compensated with the achieved success. Therefore, we investigate the test time augmentation method and introduce two separate approaches through this thesis: (1) word-level alignment of transcriptions and then constructing a new transcription out of the aligned words. (2) scoring each transcription and selecting the one with the highest score. These different approaches are detailed below.

3.4.1 Word-Level Alignment & Decoding

Handwriting images propose certain challenges that makes it hard for even people to read. Hence, it is not always possible to predict the transcription of the handwriting image without an error. We could obtain different outputs consisting of totally or partly correct predicted transcriptions, once we feed the input image along with its transformed versions. Then, we could construct a new transcription out of these outputs of the original image and of the transformed images, which could be totally correct. In this way, we could reduce the errors (if any) in the predicted transcription of the original image.

To this end, we apply 16 different shear and rotation augmentations in total to the original image during test time and retrieved transcriptions of both augmented images and of the original image. Then, we align all the transcriptions using the Myers Difference algorithm [40], which aligns two strings according to the minimum string edit distance[4]. This alignment enables us to select the most probable word at each time frame. The alignment process is illustrated in Figure 3.6 (b), where two different transcriptions are aligned to the original image transcription (Figure 3.6 (a)).

[4]We use the code implemented at https://github.com/explosion/tokenizations

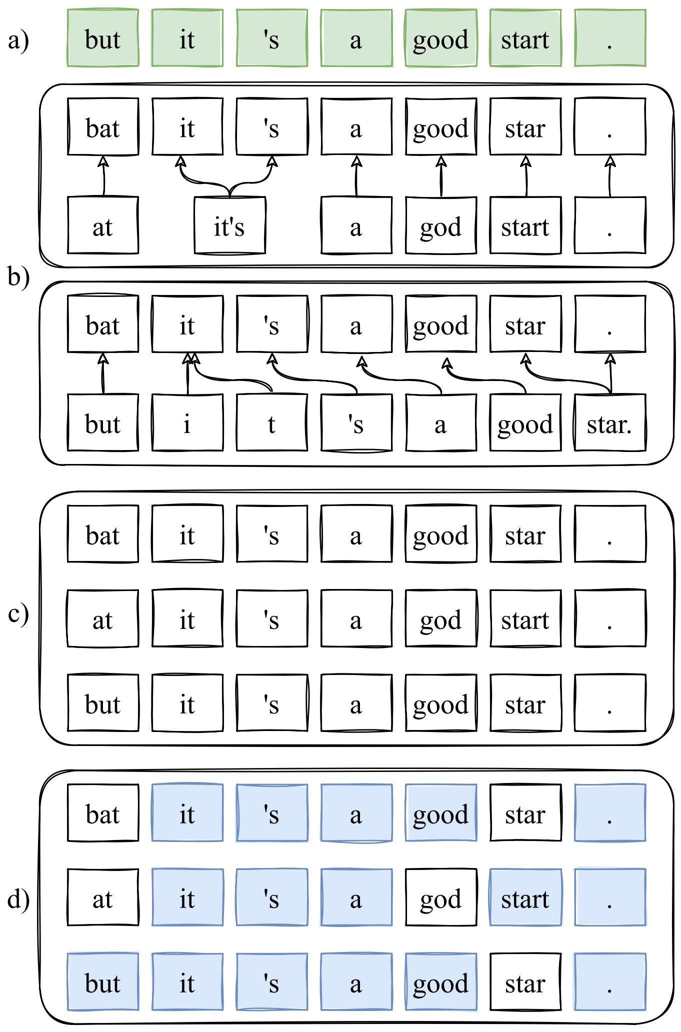


Figure 3.6 Illustration of the word-level alignment and decoding for an example line from the IAM dataset. (a) Ground truth transcription. (b) Computing alignments of two different predicted transcriptions with the original one. The sentence on the top is from the original image whereas the bottom ones are from the transformed images. (c) Transcriptions with aligned words. (d) Selecting the proper words for each time frame to build the new transcription.

Once these 17 different transcriptions (16 from augmented images and 1 from the original image) are aligned, as displayed in Figure 3.6 (c), we construct a new transcription by using two different strategies: (1) greedy and (2) beam search. In the greedy approach, we first retrieve the unique words at the current time frame. If all the words are the same, then we accept it. Otherwise, the word gets selected according to an n-gram language model, which is described below in detail. To illustrate the process:

i) first time frame of the aligned transcriptions in Figure 3.6 (d) constitute three different words: *"bat"*, *"at"*, *"but"*.

ii) these words get scored using the language model and the one with the highest score is picked, which is *"but"*.

iii) second time frame consists of the exact same word: *"it"*, which is therefore selected directly.

iv) The process continues until end of the time steps and produces a new transcription which could be equal to the original one or could be generated out of these transcriptions.

In the beam search approach, on the other hand, we extend the best k current beams with all predicted words at the next token and sort the resulting beams with respect to the 3-gram language model. This process follows the beam search idea as in [18], except that it is applied at word level. At the end, we select one transcription with the highest score among the k alternatives. We experimented with both approaches and experimentally show their effect in Section 4.5.

Language Model

We trained a 4-gram language model on WikiText-103 [38] corpus with Kneser-Ney smoothing [42] using KenLM [5] tool in order to score words or word sequences in 2-gram, 3-gram or 4-gram. Before training the language model, we applied preprocessing steps including tokenization of the words, lowering the letters and removing punctuation.

The 4-gram language model considers the likelihood of the current word given the three previous words $P(w_t \mid w_{t-3}.w_{t-2}.w_{t-1})$, as learned over the large corpus. In the example of Figure 3.6 (d), the predicted words at the first time frame position are *"bat", "at", "but"*, from which "but" is selected as it is the most likely word to start a sentence.

3.4.2 Scoring Transcriptions

Our above-defined test time augmentation method is able to craft the correct transcription from the outputs of augmented samples that might contain missing letters and/or words. However, it has drawbacks that the alignment between the transcriptions of original and of augmented images could be troublesome in some cases. Moreover, the method is composed of alignment between a sequence of words followed by the word level decoding over the aligned sentences, making the overall algorithm complex and time-taking process. Hence, we explore a simpler method in terms of running time and computational complexity.

[5]https://github.com/kpu/kenlm

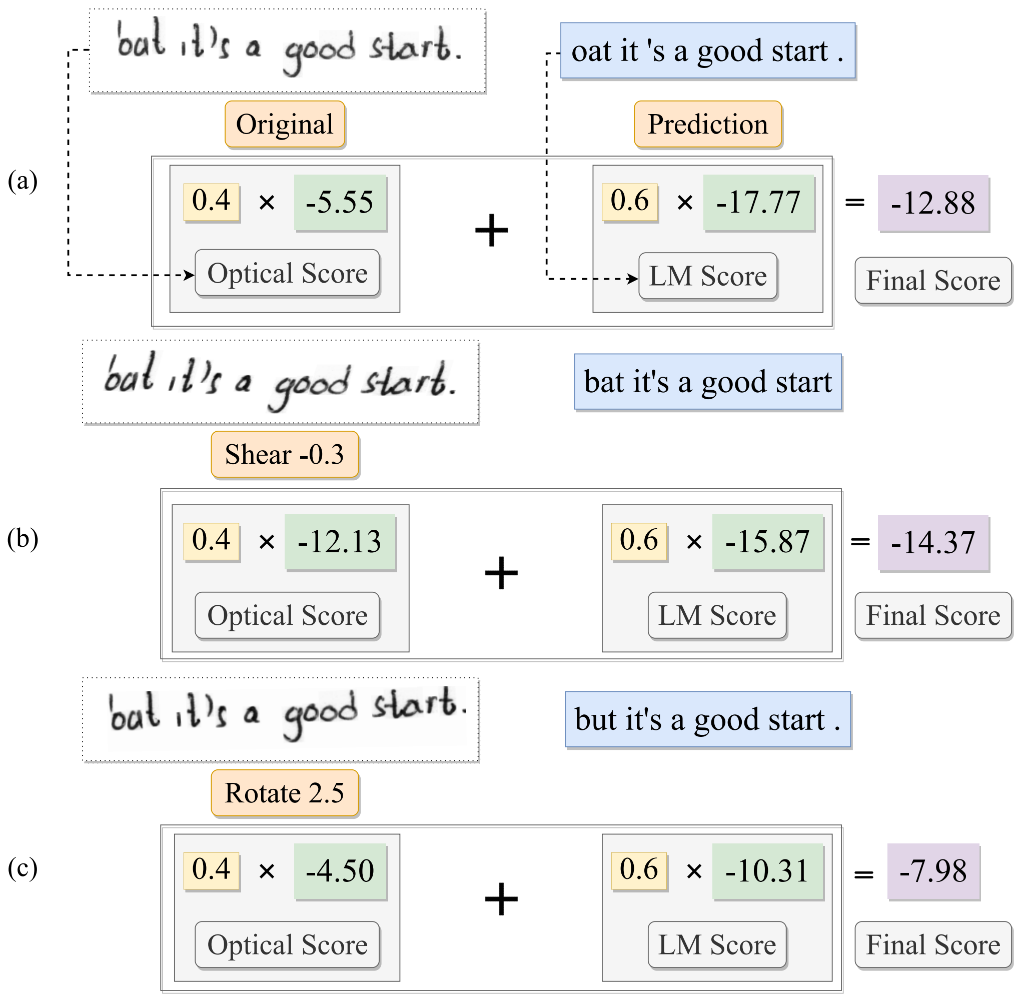


Figure 3.7 An example application of the scoring transcriptions. (a) Output of the original image is scored using the equation 3.1. The optical score is obtained from the deep learning network. The language model score is computed using the 4-gram language model. (b) and (c) shows the scoring steps of the transformed images. The output with the lowest final score is picked as the final output, which is the correct transcription of the handwriting.

In our second approach, we use test time augmentation as a rescoring method to yield a better output transcription. To this end, we apply the same transformations (16 different shear and rotation augmentations) as in the previous approach. Afterwards, we interpolate a score using an optical score of the deep learning model, OP_s, and a language model score, LM_s over the decoded transcription:

$$score = (\lambda \times OP_s) + (\omega \times LM_s) \tag{3.1}$$

Then the transcription with the highest score was picked as the final outcome. We fit values for λ and ω through the experiments.

Figure 3.7 illustrates these steps in a simplified fashion. For each image (original or transformed), the optical score, produced by the model for the image and the language model score computed for the corresponding transcription are interpolated using the above-defined formula. The best transcription with the highest score (-7.98 for the example in the figure) is picked as the final output, which is the correct decoding obtained from the image rotated by 2.5 degrees for this case.

The optical score of the deep learning model consists of probabilities getting multiplied and added through the decoding time steps by the word beam search algorithm [47]. The language model score is obtained from the same 4-gram language model of the previous approach. Yet, this time, the model is used to score the whole line, which could form a sentence or sub sentences, in contrast to scoring words or sequence of words as in the previous method.

This approach is simpler than the previous ones in terms of running time and algorithmic complexity. Further, the method achieves superior results once the best decoding is selected with respect to the lowest error, as if an oracle telling the truth. We show the effectiveness of these methods along with the *Oracle* property through the experiments in Section 4.5.

Aside from pros and cons of these two approaches, they are applicable to any model generating sequences from handwriting images. Once the real time handwriting recognition is not a must, both of these methods could be preferred considering time and computational complexity.

3.5 Application to Turkish Handwriting Recognition

The vast majority of previous works in this domain focused on English language-based recognition, as indicated before. Therefore, low-resourced languages including Turkish have attained little or no attention so far. Moreover, current works in other fields such as speech recognition and machine translation work on developing multilingual models to reduce data scarceness of low-resourced languages and increase the performance for these languages by incorporating knowledge learned from other languages. However, the same attitude has not been adopted for the offline handwriting recognition problem yet.

In this thesis, we have collected the first, to the best of our knowledge, line-based Turkish handwriting dataset. Further, we provide a baseline work using a deep learning network with other methods on the Turkish dataset for benchmarking purposes. Below we elaborate on the Turkish handwriting dataset we have crafted and share experimental results in Section 4.6.

3.5.1 Line Based Turkish Handwriting Dataset

We collected Turkish handwriting from people whose mother language is Turkish in order to: (1) propose the first, to the best of our knowledge, line-based Turkish handwriting dataset. (2) apply our method to this dataset as well to provide a baseline for future studies. (3) draw attention to low-resourced languages like Turkish.

We first determined a corpus to select Turkish text that would be hand-written by people. To this end, we found one, public Turkish language-based document repo on[6], which we will denote as Bilkent-Corpus from now on, and crafted our own Turkish text from Wikipedia. Using these documents, we collected at least one A4-page long handwritten text from *73* different people. If more than three pages were used to reflect the text, they were discarded; in other words, we included at most three pages written by the same person to stay balanced in terms of writing style. It is desired to involve more diverse handwriting styles from many more people, however, we believe that this dataset and the baseline results would be seen as preliminary attempts and encourage other people in this domain to work on low-resourced languages including Turkish.

After collecting the pages, we utilized *Xerox 3335V* printer & scanner to scan the pages into RGB formatted .png files. An example scanned page image is shown in Figure 3.11.

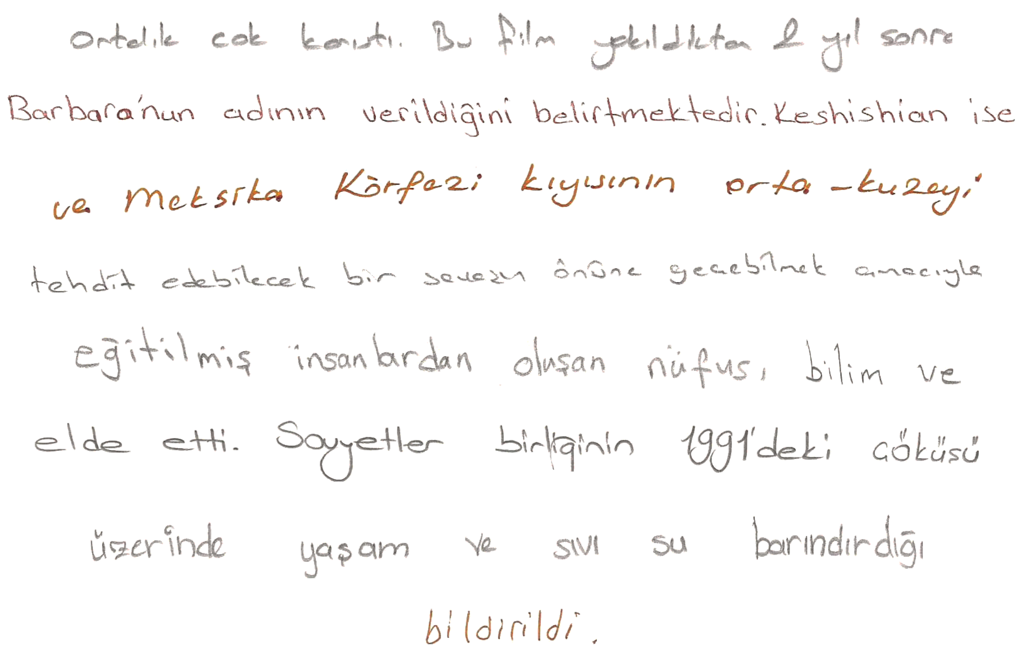

Ortalık cok karıştı. Bu film çekildikten 2 yıl sonra
Barbara'nın adının verildiğini belirtmektedir. Keshishian ise
ve Meksika Körfezi kıyısının orta-kuzeyi
tehdit edebilecek bir saldırı önüne geçebilmek amacıyla
eğitilmiş insanlardan oluşan nüfusu, bilim ve
elde etti. Sovyetler birliğinin 1991'deki çöküşü
üzerinde yaşam ve sıvı su barındırdığı
bildirildi.

Figure 3.8 Example lines from our line based Turkish handwriting dataset.

[6]https://github.com/selimfirat/bilkent-turkish-writings-dataset

Table 3.1 Number of lines and writers in train and test splits of our line based Turkish handwriting dataset.

Split	# Lines			# Writers
	Bilkent-Corp.	Wikipedia	Total	
Train	1496	434	1930	44
Test	129	582	711	29
Total	1625	1016	2641	73

Later, we cropped lines from the scanned pages using the algorithm proposed in [54]. Their method follows genetic algorithms to find line segmentation boundaries to properly crop lines from scanned pages. We used the tool [7] proposed by the authors of the paper [54]. Example cropped lines are displayed in Figure 3.8.

After cropping the lines and discarding erroneous samples, we obtained 2641 lines of which 1625 of all the lines consist of texts from Bilkent-Corpus, and the rest 1016 lines, contain texts from our Wikipedia documents. We then partitioned our dataset into train and test splits to officially denote the partitions and thereby allow future studies to make reasonable comparisons using these splits. Table 3.1 reveals necessary information about the train & test partitions.

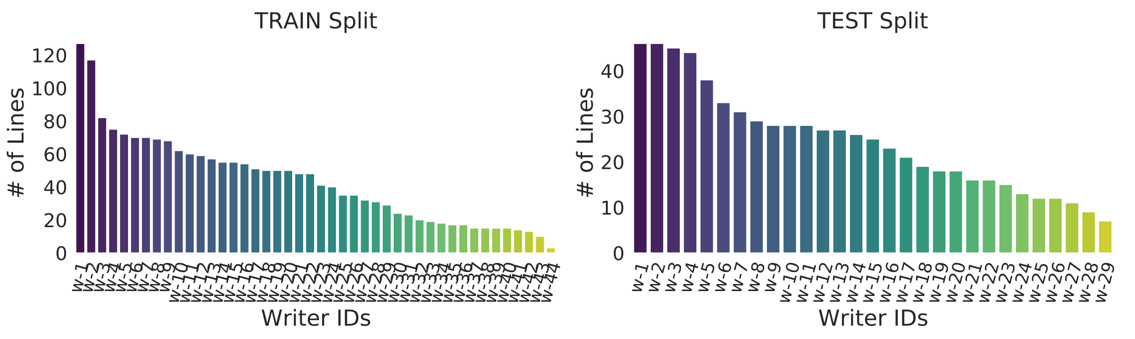


Figure 3.9 Number of lines written by corresponding writers in train and test splits of our Turkish handwriting dataset. Except the first a few writers, number of lines written by a person has similar distributions.

We paid attention not to include the handwriting of the same person in both splits at the same time; in other words, the writings of a person could be in only one partition, either only train or test. We intentionally followed this attitude to prevent predicting a learned handwriting style, which is not a possible real-life scenario. Moreover, we attempted to create two partitions having similar underlying distributions so that a model learned on the train partition could predict reasonably on the test partition, as Figures 3.9 - 3.10 display.

[7]https://github.com/GalymzhanAbdimanap/GeneticAlgorithm

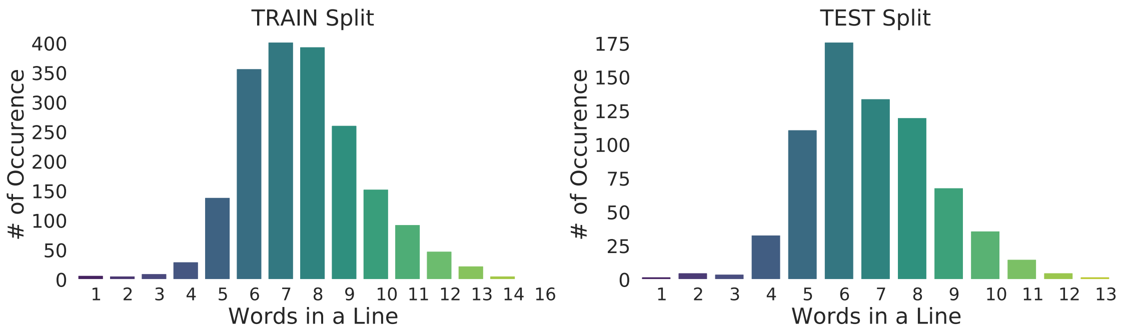


Figure 3.10 Number of words in lines of train and test splits of the Turkish handwriting dataset. In both splits lines consist of approximately similar number of words.

Figure 3.9 shows how many lines from a writer were included into the corresponding split. It is clear that there is no bias towards any writer in any of the splits, supporting our reasonable partition claim. Moreover, Figure 3.10 illustrates the distributions of number of words in lines through the splits, which are similar in both partitions as we planned.

3.5.2 Lexicon & Language Model

The Turkish language has a more complex morphological form compared to the English language; in other words, it is allowed to add suffixes to the root of a word and thereby generate a practically infinite number of words due to the agglutinative structure of the Turkish language. That is, lexicon-based approaches for Turkish handwriting recognition are not desirable since it is not possible to cover all the Turkish words and is not feasible for even the most commonly used words. However, we follow the simplest, lexicon-based approach as it is intended to provide baseline results.

We use the word beam search algorithm to decode the deep learning model output, as we did for the IAM dataset. We employ SuDer corpus as our base corpus to build the lexicon of the word beam search method. The SuDer corpus was collected for the Turkish text categorization problem (shared in[8]) in the news context and it consists of 2.7 million unique words. We further experimented with a couple of different lexicons to assess the effect of the lexicon on the Turkish language and we share the outcomes in Section 4.6.2.

[8]https://github.com/suverim/suder

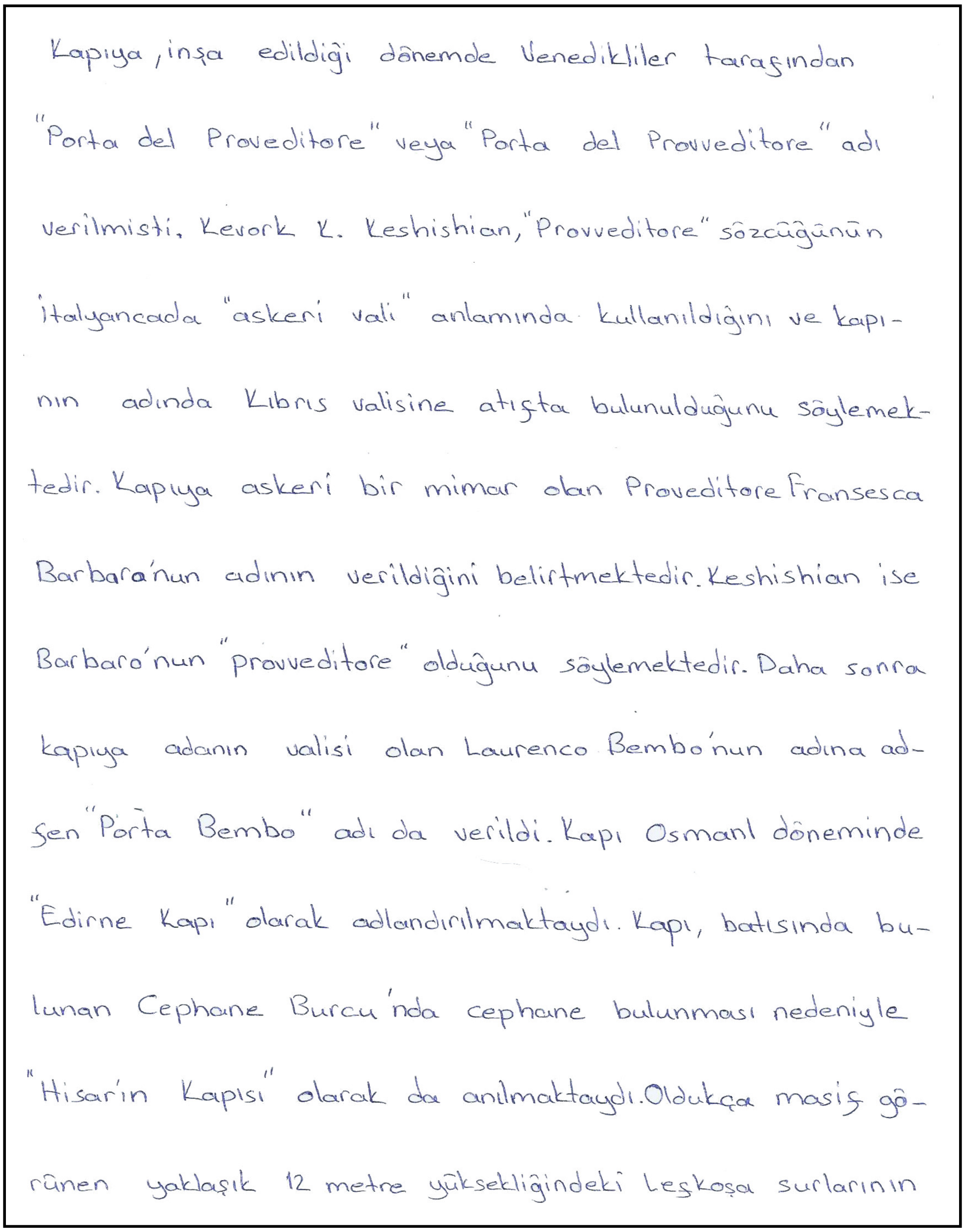

Kapıya, inşa edildiği dönemde Venedikliler tarafından
"Porta del Proveditore" veya "Porta del Provveditore" adı
verilmisti. Kevork K. Keshishian, "Provveditore" sözcüğünün
İtalyancada "askeri vali" anlamında kullanıldığını ve kapı-
nın adında Kıbrıs valisine atışta bulunulduğunu söylemek-
tedir. Kapıya askeri bir mimar olan Proveditore Fransesca
Barbara'nun adının verildiğini belirtmektedir. Keshishian ise
Barbaro'nun "provveditore" olduğunu söylemektedir. Daha sonra
kapıya adanın valisi olan Laurenco Bembo'nun adına ad-
fen "Porta Bembo" adı da verildi. Kapı Osmanl döneminde
"Edirne Kapı" olarak adlandırılmaktaydı. Kapı, batısında bu-
lunan Cephane Burcu'nda cephane bulunması nedeniyle
"Hisarin Kapısı" olarak da anılmaktaydı. Oldukça masif gö-
rünen yaklaşık 12 metre yüksekliğindeki Leşkoşa surlarının

Figure 3.11 Example scanned page containing text from our Wikipedia collection. We intentionally requested writers to give more space between lines in order to ease the job of the line cropping method we used.

4 EXPERIMENTS & RESULTS

In this section, we elaborate on the experiments we have conducted from model development to data augmentation and more. Our idea was to decide on the baseline deep learning model along with the other approaches including train time data augmentation, synthetic data generation, and decoding-based cases for the first few experiments. We employed the train and validation partitions of the IAM dataset during these experiments. After deciding on the baseline, we applied our test time augmentation methods. We experimented with both methods we proposed to assess their impact in terms of error rates, running time, and computational complexity.

We also evaluated the baseline method and our test time augmentation approach on the Turkish handwriting dataset to examine the applicability of these methods to another dataset. Moreover, we conducted a couple of experiments on the Turkish dataset such as transfer learning from the IAM and combined training with IAM and Turkish datasets in order to provide a benchmarking baseline. At last, we provide an extensive error analysis on both IAM and Turkish datasets regarding model and dataset-related issues as well as a comparison with the state of the art methods on the IAM dataset.

Training Configuration

Models were either trained with the combination of the train and validation data or only with the train data, depending on the evaluation split, for the experiments with the IAM dataset. For the Turkish dataset, the models were trained and evaluated on the corresponding train and the test partitions. Training was set to last by 200 epochs at most and stopped after 10 epochs of non-decreasing validation loss. The initial learning rate was $1e-3$ for most of the experiments though it was tailored in some cases. Rmsprop optimizer was used with weight decay being set to $1e-5$. The batch size was kept fixed at 16.

Input Image Representation

Input images were resized to 100×960 before feeding to models. It is intuitive to keep the image height fixed and resize considering the aspect ratio. However, in our preliminary experiments, we acquired worse performance using aspect ratio-based resizing. We padded images by the left and the right sides in the same batch to have the same dimensions with the aspect ratio-based resizing, which could be one of the possible reasons for the reduced performance. We tried aspect ratio-based resizing with batch size set to 1, yet this approach was neither effective nor efficient.

Decoding Methods

We employed three different decoding methods at the test phase to obtain transcriptions from model outputs, which are a sequence of probability distributions: greedy, beam search [18] and word beam search [47]. Greedy decoding is the simplest one which picks the most likely character at each time frame. Although it is intuitive to select a character with the highest probability, it is not guaranteed to generate the most likely transcription sequence.

Instead of selecting one character at a time, beam search keeps k most likely transcriptions at each time frame and produces a transcription with the highest score among these k alternatives. Word Beam Search, on the other hand, is an extension of the beam search, which constrains words to be in a given lexicon and allows non-word characters to occur inside words. Moreover, the algorithm incorporates a 2-gram word-level language model for scoring words through decoding. We set beam width as 150 and configured word beam search to utilize the 2-gram language model.

We assessed the performance of the models using character error rate (CER) and word error rate (WER), which corresponds to normalized Levenshtein Distances between the predicted and ground truth character sequences. Though we utilized the three decoding methods, we share CER and WER scores from the word beam search algorithm in the tables.

4.1 Model Experiments

Our preliminary experiments are designed to decide on the baseline model that could be used in upcoming experiments. Inspired from [8, 15, 46, 49], we followed a similar CNN + RNN + CTC architecture as explained in Section 3.1. We explored combinations of CNN and RNN models with different hyperparameters through the experiments.

4.1.1 Backbone

We first played with the number of convolution layers in the CNN backbone network. Keeping the rest of the layers fixed, we employed separate CNN networks with 8, 10, 12, and 14 layer convolutions while applying 2 or 3 max pooling operations. In addition, a smaller ResNet [23] variant, ResNet18, was utilized to experiment with a deeper model compared to previous ones. Table 4.1 shows the backbone configurations along with the obtained scores. For the sequence encoding, a two-layered bi-directional LSTM network was used with a hidden dimension of 256. During these backbone and sequence encoding experiments no data augmentation was applied to measure the direct effect of these model choices.

Table 4.1 Impact of backbone configuration on the IAM validation split.

Backbone	# Max Pooling	CER %	WER %
8 conv. layers	2	5.49	16.34
10 conv. layers	2	5.95	17.17
12 conv. layers	**2**	**5.01**	**15.14**
14 conv. layers	3	5.37	15.51
ResNet18	default	6.08	18.27

As displayed in Table 4.1, 12-layered CNN network with two max pooling operations passed the other ones. Though there is no significant performance gap, the shallower or deeper the model gets lower the scores obtained. Therefore, we preferred going with the best-scored model with the least parameters and built the baseline model on top of the 12-layered CNN network.

4.1.2 Sequence Encoding

After deciding on the backbone network, our next experiment is to select a sequence-encoder model. To this end, we evaluated the performance of bi-directional LSTM and GRU networks. We played with a different number of recurrent layers and hidden dimension sizes using these deep learning networks. Table 4.2 illustrates the model configurations and the corresponding error rates on the validation split of the IAM dataset.

Table 4.2 Performance of sequence encoding models, BiLSTM and BiGRU, with different configurations on the IAM validation split. Here the backbone network is the one with the 12 convolutional layers and 2 max-pooling operations as displayed in Table 4.1.

RNN Model	# Layers	Hidden Dim. Size	CER %	WER %
BiLSTM	1	256	4.95	15.42
		512	5.26	15.99
	2	**256**	**4.62**	**14.62**
		512	4.93	14.93
	3	256	5.07	15.64
		512	5.24	16.14
BiGRU	1	256	5.27	16.54
		512	4.90	15.48
	2	256	5.01	15.14
		512	4.85	14.94
	3	256	5.05	15.53
		512	5.70	16.33

Two recurrent layers fit better for this problem as seen in Table 4.2. Models competed with different hidden dimension sizes yet there was no exact winner of this experiment. However, we intended to keep the model simple while preserving its capabilities. Thus, we preferred to complete the baseline model with 2-layered BiLSTM having hidden dimensions of 256, which outperformed the other networks.

The rest of the experiments are built upon this baseline model consisting of 12 layered CNN with two max pooling functions and two-layered BiLSTM with 256 hidden dimensions followed by a CTC layer.

4.2 Data Augmentation

After establishing the baseline model, we explored the effect of data augmentation techniques, which are described in Section 3.3.1. The transformations were either applied separately or in combination, as shown in Table 4.3, to assess the effect of each method. We only performed one conversion on an image so as not to deform handwritings too much and it was only applied with a probability $p = 0.5$.

We determined the parameters of the transformations by visually inspecting augmented images. For the shear transformation, k is sampled from a uniform distribution between 0.6 and +0.6 with a fixed random seed to apply the same conversions through all the experiments. Rotation angle was kept in small degrees from -2.5 to $+2.5$ to mimic slanted handwriting while keeping letters inside image borders. Elastic distortion variables were picked from the following sets $\sigma \in \{3,4\}, \alpha \in \{15,20\}$. Once the parameters set, we trained the models using these augmentations and evaluated without any transformation applied.

Table 4.3 Effect of train time data augmentations applied in separate and in combination using the baseline network defined above. For the combined augmentations, only one of the transformations is used on an image to keep readability of handwritings while deforming handwriting styles. An augmentation is applied with a probability p = 0.5.

Augmentation Method	CER %	WER %
Baseline (No Augmentation)	4.62	14.62
Shear	4.28	13.44
Rotate	4.67	14.06
Elastic	4.48	14.72
Geometric	4.52	14.29
Shear + Elastic + Geometric	4.20	13.18
Shear + Rotate + Elastic	4.17	13.21
Shear + Rotate + Elastic + Geometric	**4.06**	**13.02**

Shear conversion is the most convenient way to mimic handwriting thereby the model with only shear augmentation obtained better scores compared to other single transformation applied experiments, as revealed in Table 4.3. Even though the other methods were not as effective on their own, their combination achieved better performance due to more diverse handwriting styles being presented. We obtained the best validation performance using all the transformations together. Hence, we performed the next set of experiments using these augmentations at train time.

4.3 Pretraining with Synthetic Data

Generating synthetic handwriting images is another commonly used strategy to reduce the data sparseness and increase the generalization capacity of models. Section 3.3.2 describes in detail how we produced images from the utilized corpora using an online image generation tool.

We, firstly, trained the baseline deep learning network on the generated data consisting of almost 2.5 million synthetic handwriting images. We applied the data augmentation methods, explained in Section 3.3.1, to the synthetic images in order to push the diversity further. The synthetic dataset was not split into validation or test partitions; the model was directly trained on the whole dataset by 5 epochs.

Afterwards, the pretrained network was fine-tuned on the IAM dataset train partitions in two ways: (1) freezing all the parameters of the model except the output linear layers. (2) updating all the parameters. Then, the fine-tuned model was evaluated on the IAM validation and test partitions and the corresponding scores are given in Table 4.4.

Table 4.4 Results of our pretrained deep learning model, on the IAM validation and test splits. *Linear layer*: parameters of the output linear layers are updated while fine-tuning. *All layers*: all the parameters are updated.

Pretraining	Validation		Test	
	CER %	WER %	CER %	WER %
Baseline (No Pretraining)	4.06	13.02	5.20	14.86
Pretrained:				
Linear layer	4.21	14.16	5.44	15.58
All layers	**3.88**	**12.71**	**5.05**	**14.46**

Updating only the parameters of the final linear layers did not reveal better scores, that is, presumably due to the difference in handwriting styles between synthetic and natural ones. Hence, we experimented with adapting all the parameters for real handwriting images. As reflected in Table 4.4, pretraining on synthetically generated handwriting images and then fine-tuning over natural images boosted the performance.

We reduced the character errors by around 0.15% and word errors around 0.4% (from 5.20% to 5.05% for character errors and from 14.86% to 14.46% for word errors) by only pretraining the model on more than 2 million synthetic handwriting images. However, there is room for improvement once there is more synthetic and natural handwriting data for pretraining and fine-tuning. We performed the next set of experiments using this pretrained model.

4.4 Effect of Lexicon, Letter Case, and Punctuation

We decoded model outputs using three different methods (greedy, beam search, and word beam search) and indicated that the results on the tables are from the word beam search algorithm, as described in the beginning of this chapter. We used greedy and beam search methods as is; in other words, without adding an external language model or considering a lexicon during decoding. However, word beam search builds a prefix tree from a corpus i.e. lexicon to keep decoded words in the lexicon. Therefore, we explored the effect of lexicon size and out-of-vocabulary rate on decoding performance.

We experimented with three different English corpora and their combinations for building the lexicon. The base lexicon consists of the texts from the IAM training partition, which is part of the LOB corpus [1]. On top of it, we appended the Brown corpus, which is a million-word English text corpus consisting of texts from 15 different categories. We also added the WikiText-2 corpus comprising over two million English words. Further, we assessed their combination as well as the validation partition of the IAM dataset. The lexicon built upon the IAM validation split has zero out-of-vocabulary rate, which is not possible for real-time scenarios yet it underlines the effect of the lexicon.

Table 4.5 Reflection of lexicon size with number of unique words (after tokenization, lowering and discarding punctuation) and out-of-vocabulary (OOV.) rates on errors. B stands for the Brown corpus and W2 for the WikiText2. corpus

Lexicon	# Words	OOV. Rate	CER %	WER %
Baseline (IAM Train)	7.845	0.54	3.88	12.71
+ B	57.606	0.15	3.49	11.77
+ W2	60.244	0.21	3.56	11.92
+ B + W2	**82.449**	**0.13**	**3.46**	**11.70**
+ IAM Validation	2.251	0.00	2.2	6.35

Table 4.6 Effect of the letter case and the punctuation when decoding over the validation split of the IAM dataset. Decoding method up to this experiment involved case sensitive letters with punctuation.

Case & Puncutation	CER %	WER %
Baseline (Case Sensitive + Punctuation)	3.46	11.70
Case Insensitive + Punctuation	3.16	10.58
Case Insensitive + No Punctuation	**2.87**	**8.27**

The error rate depends on the context of the words in the lexicon. As the rate of the out of vocabulary (OOV) words decreased, the error rates gets reduced as well, as illustrated in Table 4.5. In a normal test scenario, it is not possible to contain all likely words in a lexicon, which is the IAM Test case in the table. Nonetheless, most of the time, bigger the lexicon is lower the errors achieved. However, one needs to account the trade-off between the running time with a bigger lexicon and obtained performance. We preferred going with the lexicon consisting of train partition of IAM, Brown and WikiText-2, since there was no significant difference of running time compared to smaller lexicons in Table 4.5.

We decoded and compared the transcriptions with the ground truth text considering case-sensitive letters with punctuation. However, some denoted that they generated case insensitive text without any punctuation [15, 58] while others even did not inform about the decoding method. Hence, we performed experiments considering both strategies.

The IAM dataset contains sentences with several quotes and punctuation around them where missing those signs is quite possible due to their adjacent written style. Moreover, the predicted transcription could be the lower-cased or upper-cased of the same letter though this is rare. Thus, considering letters insensitive without any punctuation reduces the error considerably, as revealed in Table 4.6. However, unless otherwise stated explicitly, we shared all the errors measured with case sensitive letters and punctuation through the tables.

4.5 Test Time Augmentation

Test time augmentation is a simple and effective technique with allowable time complexity for offline handwriting recognition. We presented our two different test time augmentation approaches in Section 3.4, and here we demonstrate effectiveness of our test time augmentation methods along with their pros & cons.

Basically, for the first method: (1) we utilize the same shear and rotation transformations of the training phase, 8 different conversions for each, at the test time. (2)

obtain 17 transcriptions in total (16 from transformations, 1 from the original). (3) align the transcriptions with the original output using Myers Difference algorithm [40]. (3) pick words at each time frame considering the 4-gram language model, described in Section 3.4, to obtain the final transcription. We either pick one word at each time frame as a greedy approach or keep $k = 10$ most probable words at each time frame as a beam search approach and produce the highest scored transcription. Corresponding error rates are reflected as the greedy and the beam search in Tables 4.7 - 4.8.

However, for the second method, we prefer a simpler way and follow the above-defined first two steps. Once we attain the transcriptions of the transformed images and of the original image, we score the transcriptions using the formula, defined in Section 3.4.2, and chose the one with the best score.

Once the final transcription is obtained, it was compared with the ground-truth transcription and the error rate in terms of CER and WER metrics was measured. We assessed the proposed test time augmentation methods on both case sensitive letters with punctuation and case insensitive letters without punctuation to give better insight.

Table 4.7 Validation and test scores with our two test augmentation methods considering **case sensitive** letters **with punctuation**. Word Level Alignment indicates the scores from the first test time augmentation approach. *Combined Scoring* denotes the scoring function that aggregates optical and language model scores, defined Section 3.4.2. *Oracle* indicates the case in which transcriptions are obtained with respect to the lowest character error.

Test Time Augmentation	Validation		Test	
	CER %	WER %	CER %	WER %
Baseline (No TTA)	3.46	11.70	4.80	13.85
Word Level Alignment				
Greedy search	3.34	11.02	4.61	12.94
Beam search	3.28	10.91	4.51	12.60
Scoring Transcriptions				
Combined Scoring	3.22	10.63	4.37	12.03
Oracle	2.28	8.54	3.02	9.35

Table 4.7 shows that applying test time augmentation reduced the character errors by almost 0.5% and word errors by around 2%, when decoding with case-sensitive letters and punctuation. It is reflected through the table that our second method, *Scoring Transcriptions*, outperformed the first approach with less complexity and

significantly less running time, which is revealed in Table 4.9. This is presumably due to alignment problems that might occur between the transcriptions, which led to these bigger error rates. However, it is seen that keeping more than one word at a time beats the greedy approach where only the most probable word is chosen. The error rate gets decreased further (almost 1% for character errors and 2.5% for word errors) when decoding with case insensitive letters without punctuation, as displayed in Table 4.8.

In addition to combined scoring function, we performed test time augmentation and selected the final transcription with respect to the lowest CER score. We named this method as *Oracle* since it knows in advance the error due to the comparison with the ground truth sequence, which is not a possible real scenario. However, we aimed to show how much gain is possible with this *Oracle* method. As Tables 4.7 and 4.8 indicate, there is room for improvement once a better scoring function is used.

Table 4.8 Validation and test scores with our two test augmentation methods considering **case insensitive** letters **without punctuation**.

Test Time Augmentation	Validation		Test	
	CER %	WER %	CER %	WER %
Baseline (No TTA)	3.38	11.29	4.38	12.00
Word Level Alignment				
Greedy search	3.09	9.86	4.10	10.07
Beam search	3.01	9.74	3.99	9.98
Scoring Transcriptions				
Combined Scoring	2.84	9.05	3.59	9.44
Oracle	1.61	6.42	2.63	7.74

Computation & Running Time Complexity

We also analyzed the time complexity of our proposed methods and compared running time with the decoding of original images. It was not straightforward to compute the algorithmic complexity of our first method due to the alignment step. However, it is observed that the first method is computationally more complex compared to our second method whose complexity is defined below. For our second method, the computational complexity was merely based on the complexities of the 4-gram language model and the word beam search algorithm which is described in [47]. The time complexity to decode one input sample as computed in [47]:

$$\mathcal{O}(T \times BW \times C \times (\log(BW \times C) + M + \log(W)))$$

In this Big-O notation, T denotes generated sequence length, BW stands for beam width, C for maximum number of recognizable characters, M for maximum length of a word, and $\log(W)$ is the time for a lookup table. In our case, due to 17 total sample decodings and language model based scoring of these transcriptions, the complexity become:

$$\mathcal{O}(17 \times T \times BW \times C \times (\log(BW \times C) + M + \log(W)) + (17 \times LM))$$

The running time of obtaining the final transcription was increased as we processed 16 more images. We run our proposed methods on two different machines with the following configurations: (1) AMD Ryzen 3970X, 64 GB ram with Geforce RTX 3090. (2) Intel i7 10700K, 32 GB ram with Gefore RTX 3080. The time taken for each image was measured and its average with the standard deviation for all the samples in the IAM test partition is computed for both of our methods, as shared in Table 4.9.

The table illustrates that applying the test time augmentation method slows down obtaining the final transcription. However, we reduced the running time and computation complexity of our first approach when replaced with the second one, as Table 4.9 displays. Considering the increased running time and decreased error rate with our second test time augmentation method, it could be preferred for offline handwriting cases when there are no real-time requirements. Additionally, a better scoring function would reveal greater success without needing any more model parameters or more data as *Oracle* row indicates the room for such ıimprovement in Tables 4.7 and 4.8.

Table 4.9 Running times of our test time augmentation methods on two different machines with CPU & GPU options. Mean and standard deviation of the elapsed time over the IAM test partition. *1st Method* denotes our method Word Alignment & Decoding and *2nd Method* indicates the Scoring Transcriptions method.

Machine	Processing Device	Original Time (s)		1st Method (s)		2nd Method (s)	
		Mean	Std.	Mean	Std.	Mean	Std.
PC 1	CPU	2.58	1.03	45.52	5.21	21.46	4.08
	GPU	2.27	0.98	41.27	5.02	19.33	3.84
PC 2	CPU	3.64	1.12	49.88	6.56	25.54	4.56
	GPU	3.28	1.02	46.54	6.24	24.76	4.21

4.6 Offline Turkish Handwriting Recognition

In the previous experiments, we focused on our baseline method including the deep learning network, data-relevant approaches, and our proposed test time augmentation method. All these experiments were conducted on the IAM dataset to evaluate the effect of each experiment properly. We further explore the applicability of our method on another dataset which is our line-based Turkish handwriting dataset, introduced in Section 3.5, and we provide baseline results for future studies.

4.6.1 Comparison of Methods

First, we pretrained our baseline model on our synthetic dataset and fine tuned on the IAM dataset with a new alphabet consisting of Turkish characters. Then, we directly evaluated the model on the test split of our Turkish dataset and the corresponding error rates were reflected in the first row of Table 4.10. The model made a large number of mistakes predicting characters and thereby words, which is supposedly due to unobserved Turkish characters that were not presented to the model at training.

Our next experiment was to train and evaluate the model on the corresponding splits of our Turkish dataset to assess the importance of Turkish data in training. The model achieved superior results compared to the previous one as shown in the second row of Table 4.10. Even though the model was trained with far less data (2641 from Turkish vs. 6161 from IAM), error rates got reduced remarkably (from 15.21% to 6.01% for CER and from 49.37% to 25.02% for WER), which points to the significance of Turkish handwriting being used at training.

Table 4.10 Evaluation results on our Turkish dataset. Method describes how the deep learning networks are trained and/or fine tuned. *TR* denotes our line based Turkish handwriting dataset.

Method	CER %	WER %
Trained on IAM	15.21	49.37
Trained on TR	6.01	25.92
Trained on IAM. Fine tuned on TR	**4.28**	**20.29**
Trained on IAM + TR	4.42	20.96
Pretrained on synth	6.18	27.70

Afterwards, we fine tuned the first model on the Turkish dataset to make use of synthetic and IAM data as well as Turkish handwriting. Table 4.10 illustrates that transferring knowledge learned on the synthetic and the IAM datasets by fine tuning on the Turkish dataset beat the previous approaches substantially (from 6.01 % to 4.61 % for CER and from 25.92 % to 21.16 % for WER). That is, incorporating information from both English and Turkish natural handwriting plus synthetic data is a promising approach.

To further reduce the error rates, we trained the model on a combined dataset consisting of train splits of IAM and Turkish datasets. We obtained similar error rates to the previous experiment, emphasizing the former claim of including IAM and Turkish datasets in the training of models. Additionally, we directly evaluated the model that was pretrained on the synthetic dataset, which is the last entry of Table 4.10. This one also reflects the importance of Turkish data usage at train time, which is consistent with the previous experiments.

4.6.2 Test Time Augmentation

We applied our test time augmentation methods to the IAM dataset and showed the effectiveness of our second approach, as described in Section 4.5. We perform our second test time augmentation technique on the Turkish dataset to further assess the method by evaluating it with another dataset.

Language Model We trained a 4-gram language model for the Turkish language to be utilized with the test time augmentation method. We followed the same language model configuration of our English based language model. That is, we used Kneser-Ney smoothing [42] to diminish the effect of out of vocabulary words and applied the same preprocessing steps considering the Turkish language. We combined the Bilkent-Corpus and the SuDer corpus to train the language model using the KenLM tool.

We employed the best-scoring model on the Turkish dataset, which was pretrained with synthetic data, fine tuned on the IAM dataset, and then fine tuned on the Turkish data as well. Table 4.11 shows that our test time augmentation approach revealed lowered error rates compared to the baseline model. The amount of error reduction, 0.3% CER and around 2% WER (from 4.28% to 4.03% for character errors and from 20.29% to 18.60% for word errors), is similar to the case of the IAM dataset, which was depicted in Table 4.7.

Table 4.11 Error rates obtained with our second test time augmentation method when applied on the Turkish dataset using the best model.

Test Time Augmentation	CER	WER
Baseline (No TTA)	4.28	20.29
Scoring Transcriptions		
Combined Scoring	4.03	18.60
Oracle	2.82	14.72

Moreover, *Oracle* approach obtained better error rates indicating possible improvement with the test time augmentation method. These results support our claim of using test time augmentation and empirically show the effectiveness of the method on two different datasets with two different languages.

4.6.3 Lexicon Effect

We empirically showed that incorporating Turkish data into training reduced the errors remarkably, as illustrated in Table 4.10. However, lexicon also plays an important role for the Turkish based handwriting recognition as explained in Section 3.5.2. Therefore, we experiment with different lexicons to measure the importance of the lexicon for the Turkish language.

We constructed our base lexicon from the SuDer corpus as described in Section 3.5.2. The corpus contains 2.7 million unique words and 11% of the words in the test split of the Turkish dataset are not present in the SuDer corpus. Since the out-of-vocabulary rate is relatively low, we obtained modest error rates as revealed in Table 4.12.

In order to evaluate the effect of the lexicon, we tried the Bilkent-Corpus as the lexicon, which contains far less unique words compared to the SuDer corpus. Bilkent-Corpus consists of almost 290k unique words while out of vocabulary rate is equal to 22%. Table 4.12 illustrates the significant change of the error rates between the use of SuDer and Bilkent-Corpus lexicons.

Table 4.12 Illustration of the lexicon effect on Turkish dataset. *Bilkent and Wikipeda* corpora are described in Section 3.5.

Lexicon	# Words	OOV. Rate	CER	WER
Bilkent-Corpus	289.936	0.22	5.75	25.36
Base Lexicon (SuDer)	2.770.813	0.11	4.28	20.29
SuDer + Wikipedia Test	**2.771.140**	**0**	**3.39**	**16.30**

We employed the text from the test split of our Wikipedia corpus to build the lexicon with zero out of vocabulary rate, demonstrating the maximum possible gain with a proper corpus. We obtain the best error rates once the lexicon contains all the words in the test set, which is depicted in the last row of Table 4.12. As these results suggest, the lower the out-of-vocabulary rate is better the performance achieved, even for this type of lexicon-based Turkish handwriting recognition. However, advanced approaches including lexicon-free or sublexical unit-based decoding as proposed in [60] could reduce the error rates further, which we left for future works.

4.7 Error Analysis

Up to this point, we measured the errors in terms of CER and WER metrics and computed their mean over the test partitions of the IAM and the Turkish datasets. Once we examined the erroneous cases we found out that assessing the performance by only looking at average CER/WER scores could be lacking. Hence, we provide analysis in terms of cumulative errors, presented in Figures 4.1 and 4.3, explore hard samples along with possible ways to overcome these challenging handwriting samples, and share mislabelled example lines from the IAM dataset.

Our intention is to make a detailed analysis using our best-performing models on both IAM and Turkish datasets. We preferred going with the outputs of the best models without the test time augmentation method for the error analysis. However, it is possible to explore the errors using the test time augmentation outputs which would reveal similar results.

Table 4.13 Mean and standard deviation of number of characters and words in lines of the test splits.

Dataset	#Lines	#Characters per Line		#Words per Line	
		Mean	Std.	Mean	Std.
IAM Test	2915	35.20	8.72	8.90	2.69
Turkish Test	698	45.54	10.56	6.88	1.78

Errors on IAM Dataset

We computed character and word error rates in a cumulative form and observed the following points. In 42% of all the lines, all the characters were predicted correctly as illustrated in Figure 4.1. A more striking observation is that the model incorrectly predicted at most two characters in 72.25% of all the images in the IAM test set, which has an average of 35.2 characters per line image with an 8.72 standard deviation as indicated in Table 4.1 (a). The figure further tells that in 91.22% of all the samples the model made at most five character mistakes.

Moreover, it is seen in Figure 4.1 (b), the model produced at most two incorrect words in almost 84.49% of all the samples in the test split which holds a mean of 8.90 words per line with standard deviation of 2.69. These results suggest that inspecting cumulative errors along with the average metrics would reveal better insights.

Handwriting images could be messy that even some might struggle reading what is written on an image, as illustrated in Figure 4.2. The figure shows examples from IAM test dataset that our model predicted at least five characters incorrectly i.e. CER $>$ 5, corresponding to 7% of all the lines. Figure 4.2 demonstrates that reading

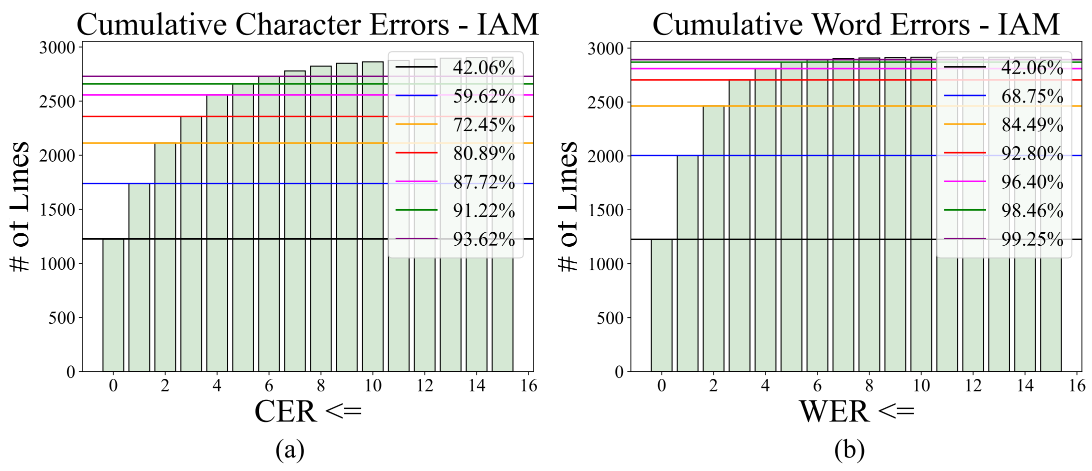
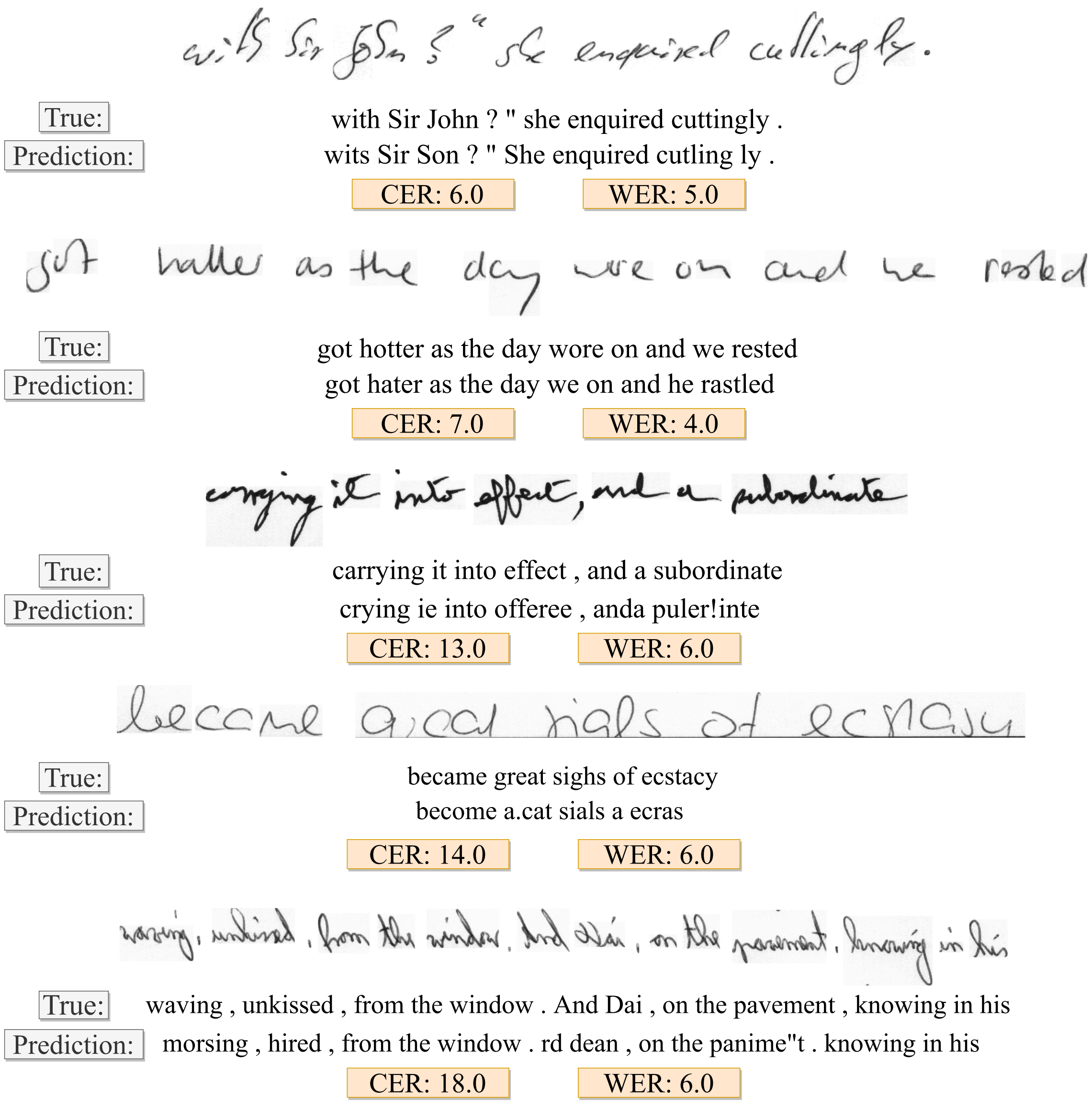


Figure 4.1 Cumulative character and word error rates obtained with the baseline model on the IAM dataset. (a) Illustrates cumulative character errors. Example: All the characters are predicted correctly in 42% of all the lines, which corresponds to 1226 line images; one or at most two characters are predicted incorrectly in 560 line images, which corresponds to 17% of all the lines (b) Displays cumulative word error rates following the same manner.

these handwriting text is not as easy for people too. That is, state of the art deep learning models with advanced techniques would probably miss theses cases as well. Thus, it is better to measure this type of cumulative character and word errors in addition to the averaged errors.

Furthermore, the IAM dataset contains partially or totally wrong ground truth labels which, in fact, affects the both training and testing phases. Therefore, we found and corrected the erroneous cases on the IAM test split in order to evaluate better. Example erroneous lines along with their issues are presented in the Appendix A. We will share the new, fixed metadata of the test split on the link [1].

[1]https://github.com/firatkizilirmakk/handwriting-recognition


True: with Sir John ? " she enquired cuttingly .
Prediction: wits Sir Son ? " She enquired cutling ly .
CER: 6.0 WER: 5.0

True: got hotter as the day wore on and we rested
Prediction: got hater as the day we on and he rastled
CER: 7.0 WER: 4.0

True: carrying it into effect , and a subordinate
Prediction: crying ie into offeree , anda puler!inte
CER: 13.0 WER: 6.0

True: became great sighs of ecstacy
Prediction: become a.cat sials a ecras
CER: 14.0 WER: 6.0

True: waving , unkissed , from the window . And Dai , on the pavement , knowing in his
Prediction: morsing , hired , from the window . rd dean , on the panime"t . knowing in his
CER: 18.0 WER: 6.0

Figure 4.2 Hard samples from the IAM dataset test partition, illustrating lines for which our model predict more than five characters incorrectly (i.e. CER > 5).

Errors on Turkish Dataset

We performed the same error analysis for the Turkish dataset as well. Cumulative error rates in Figure 4.3 reflect that the model was able to predict almost 33% of all the test samples in the Turkish dataset without an error. Moreover, the model produced at most two incorrect characters in 68.19% of all the lines. On the other hand, 80% of all the predictions contain one or two word errors as showed in Figure 4.3 (b).

Since we obtained modest performance on the Turkish dataset, as shown in Table 4.10, the cumulative errors reveal similar outputs too. In addition to the cumulative analysis, we inspected samples that the model attained high error rates. We observed that the most of the handwriting is clear and readable for the cases with at least five

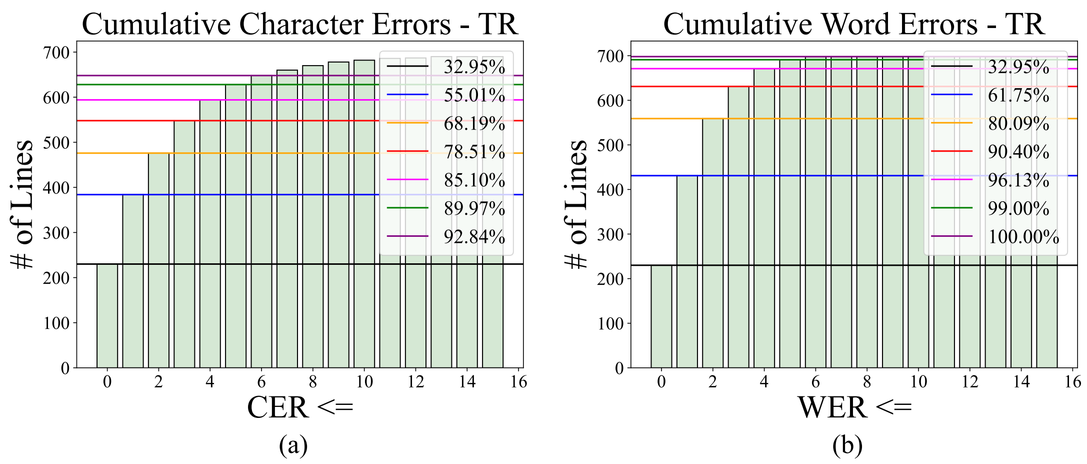
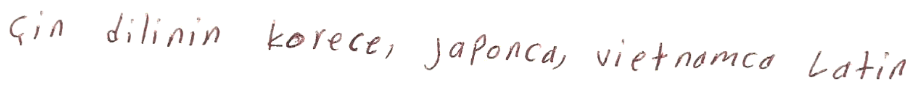


Figure 4.3 Cumulative character (a) and word (b) error rates obtained with the baseline model on the Turkish dataset.

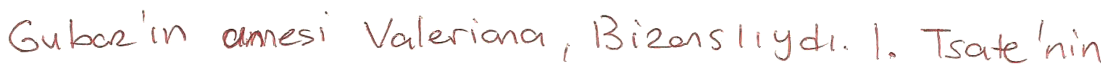
incorrectly predicted characters. More specifically, people whose mother tongue is Turkish can read most of these handwriting images where the model struggle with. Figure 4.4 displays example lines that we consider as hard samples since the model obtained at least five character errors i.e $CER > 5$.

There are a couple of reasons for these results. First and the most important reasons is the lack of sufficient amount of Turkish handwriting data, which could be used to train models. As we demonstrated through the previous experiments on the Turkish dataset, more the Turkish handwriting is available lower the errors obtained. Therefore, increasing amount of data could reduce error rates considerably.

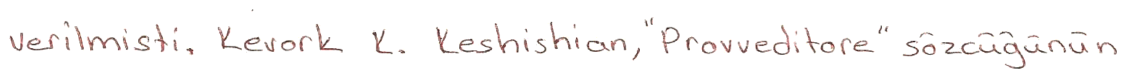
Furthermore, another important reason of these relatively high errors is the lexicon we used to decode words from. For agglutinative languages like Turkish, using a lexicon to select words from is not the best way of decoding model outputs. Therefore, lexicon-based approaches for the Turkish handwriting recognition are likely to produce partly or totally incorrect words. The last example line in Figure 4.4 contains a word "*ulaşmadan*", for which our model produced the word "*ulaşmada*", which is wrong. Even though these words hold almost the same meaning, they are different and thereby the word error gets increased by one. Additionally, the lines consisting of text from our Wikipedia corpus might contain terminological words, phrases or custom names, as displayed in Figure 4.4, which makes the lexicon-based decoding even harder.



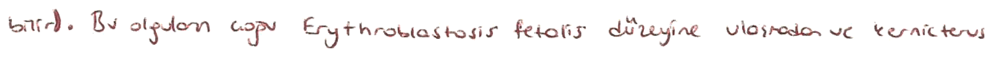
True: Çin dilinin Korece, Japonca, vietnamca Latin
Prediction: çin dilinin korece, sapanca vietnamca katin
CER: 6.0 WER: 4.0



True: Gubaz'ın annesi Valeriana, Bizanslıydı. I. Tsate'nin
Prediction: Gubaş'ın annesi Valerian, Bizans'ıydı. I. Tsatsarov
CER: 8.0 WER: 4.0



True: verilmişti. Kevork K. Keshishian, "Provveditore" sözcüğünün
Prediction: verilmişti. Kevork K. Keshi5hin, proveke5e" sözcüğünün
CER: 10.0 WER: 2.0



True: bilir). Bu olguların çoğu Erythroblastosis fetalis düzeyine ulaşmadan ve kernicterus
Prediction: birl. Bu olgular çoğu Erythrai5tasis feta!is düzeyine ulaşmada ve keni5ters
CER: 16.0 WER: 5.0

Figure 4.4 Samples where our deep learning network makes more than five character errors (i.e. CER > 5).

4.8 Comparison with the State of the Art

In this section, we explore the state of the art methods regarding architecture, data and other techniques they incorporated into. Our proposed methods obtain lower scores compared to the current state-of-the-art studies as displayed in Table 4.14. There are three main reasons of this drawback: (1) Advanced deep learning architectures with great image processing and sequence learning capabilities, (2) Internal or synthetic data, (3) External language model and a lexicon.

However, direct comparison of the methods where the models trained on different datasets would not yield the most accurate comparison. Yet, our model is able to compete with the similar approaches such as the one proposed by Xiao et. al. [59]. Here the underlying difference could be due to the quality or amount of the synthetic data as well as using LOB [1] corpus for the word beam search lexicon.

Others exploited the attention based approaches to make use of their advantages on sequence learning. In addition to deep learning architectures, they generated millions of synthetic handwriting images and/or crafted real handwriting images through the web, which is another factor of their success. As summarized in Table

4.14, Li et. al. [35] pretrained a Transformer model on their synthetic dataset, finetuned over the IAM and achieved superior results without applying any post processing method e.g. a language model or a lexicon. Diaz et. al. [13] obtained the state of the art performance where they preferred a simpler model with considerably less amount of parameters, trained on datasets they collected and supported the outputs with a 9-gram language model.

These works suggest that the crucial part of a successful handwriting recognition system is to utilize large number of high quality data, both synthetic and real. Once the data requirement is satisfied, it is better to employ the state of the art deep learning models such as Transformer, Vision Transformer or other models with attention mechanism. Next, the final output could be decided with the help of an n-gram language model to further increase the success.

Table 4.14 Comparison of our approach with the state-of-the-art approaches. LM & Lexicon denotes the language model and the lexicon being used (if any) during decoding. Decoding indicates which method is used to decode network outputs into character sequence. Here WBS denotes the word beam search algorithm and *bw* is the beam width. #P is the number of trainable parameters (in millions) of the deep learning networks. CaseInsens. stands for case-insensitive decoding and NoPunct. denotes the decoding without considering punctuation.

Authors	Encoder	Decoder	Train Set	LM & Lexicon	Decoding	#P	CER %	WER %
Bluche [8]	GCRNN	CTC	IAM + Multi.	7-gram LM + 50K	-	0.75	3.2	10.5
Xiao [59]	CNN + LSTM	CTC	IAM + Synth.	Brown + LOB	WBS bw/150	-	3.03	8.66
Michael [39]	CNN + LSTM	LSTM w. Att.	IAM	-	Beam bw/16	-	4.87	-
Kang [29]	Transformer		IAM + Synth.	-	Greedy	100	4.67	15.45
Li [35]	Transformer		IAM + Synth.	-	Beam bw/10	334	3.42	-
Li [35]	Transformer		IAM + Synth.	-	Beam bw/10	558	2.89	-
Diaz [13]	Self Att. + CTC		Public	9-gram LM	Greedy	~12.5	3.15	-
Diaz [13]			Internal + Public	9-gram LM	Greedy	~12.5	2.75	-
Ours	CNN + LSTM	CTC	IAM + Synth.	Brown + W102	WBS bw/150	~3.6	4.80	13.85
					WBS + TTA		4.37	12.03
					WBS + TTA + CaseInsens. + NoPunct.		3.59	9.44

5 CONCLUSION & FUTURE WORK

Summary

In this thesis, we have studied the offline handwriting recognition problem using deep learning networks with data-oriented approaches. We made an extensive analysis using a CNN + BiLSTM + CTC network architecture on the line-level IAM dataset. We experimented with the components of the network to decide the baseline model and showed the impact of each component on the recognition performance.

Following the literature, we applied train time data augmentation and pretrained the network in order to diminish the effect of data sparseness. On the augmentation side, we found that the shear transformation is more effective than the other conversions. However, the effectiveness of data augmentation is pushed further when all the defined conversions are included. In addition, we generated more than 2 million synthetic handwriting line images to pretrain the network and thereby alleviate the data scarcity. We showed that pretraining the model with around 2 million synthetic images and then fine-tuning with the real images of the IAM dataset revealed a better recognition performance; errors were reduced by 0.5%. However, incorporating more synthetic data into the training, as done in [13, 35], could reduce the error rates further.

The test time augmentation method has not been explored in detail in this domain. This is especially due to the increased running time and computational complexity of the overall handwriting recognition system. However, the idea is simple and could be effective if it is used properly. Therefore, we proposed two different test time augmentation approaches and demonstrated their success in the recognition performance along with their pros cons. Though both of these methods reduce the error rates, our second approach, named as scoring the transcriptions, outperforms the first one with less complexity and running time. While our second method decreased word errors by 2.5%, it is possible to drop the errors further with a better scoring function as we pointed to the *Oracle* property.

Previous works in this domain have mostly focused on the English language-based approaches. Therefore, we intended to make the first steps of line-based offline handwriting recognition for the Turkish language. To this end, we created a line-level Turkish handwriting dataset, which will be the first, public dataset to the best of our knowledge. Moreover, we performed experiments on this dataset with our method including the deep learning network and data-related techniques to provide comparable, baseline results for future studies.

The amount of Turkish handwriting data is not sufficient to obtain great results with deep neural networks. Therefore, we followed such approaches as transfer learning and joint training to leverage Turkish handwriting recognition by utilizing the knowledge from the English language. We uncovered that transferring the learned knowledge from the IAM to the Turkish dataset reduced the error rates remarkably compared to models trained only on the IAM or on the Turkish dataset. Further, we experimented with different lexicons and applied our test time augmentation method to this dataset. Our method obtained lower error rates as in the IAM dataset, showing the effectiveness of the approach on two different language-based datasets.

After all these works, we provided a comprehensive error analysis on both datasets. We illustrated through the cumulative error rates that the model recognized 42% of the IAM test split without an error and made 1 or 2 character errors in 30% of the rest. In other words, our model made a maximum of 2 character errors in 72.45% of the set in which the average number of characters per line is 35, which was a striking observation. However, for the Turkish dataset the errors are caused mainly due to lack of sufficient amount of train data and the lexicon being used.

At last, we made comparisons of our method with the state-of-the-art approaches on the IAM dataset. We justified the reason of being passed by these methods as following: (1) our deep learning network is relatively simple compared to the advanced Transformer architectures. (2) we incorporated far less synthetic and natural handwriting images.

Future Directions

We left applying the current state-of-the-art approaches such as Transformer and Vision Transformer networks in combination with a large number of real and synthetic handwriting images for future work. Moreover, methods involving self-supervised and unsupervised techniques as in speech recognition field [3, 12, 28] in conjunction with multilingual handwriting recognition seem promising and remain to be explored. Finally, sharing the collected data with researchers will lead to a more accurate comparison of methods in the field and a better solution to the problem.

BIBLIOGRAPHY

[1] Lancaster-oslo-bergen corpus of modern English (LOB) : [tagged, horizontal format] / stig johansson. Oxford Text Archive.

[2] Aydemir, M. S., Aydın, B., Kaya, H., Karlıağa, , & Demir, C. (2014). Tübitak Turkish — Ottoman handwritten recognition system. In *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, (pp. 1918–1921).

[3] Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations.

[4] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate.

[5] Bahl, L., Brown, P., de Souza, P., & Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, (pp. 49–52).

[6] Bartos, G. E. (2021). Deep learning based offline handwritten character recognizer systems with a multilingual handwritten character dataset.

[7] Bilgin-Tasdemir, E. & Yanikoglu, B. (2018). Large vocabulary recognition for online Turkish handwriting with sublexical units. *Turkish Journal of Electrical Engineering and Computer Sciences*, *26*, 2218 – 2233.

[8] Bluche, T. & Messina, R. (2017). Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, (pp. 646–651).

[9] Bunke, H., Bengio, S., & Vinciarelli, A. (2004). Offline recognition of unconstrained handwritten texts using hmms and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 709–720.

[10] Çapar, A., Taşdemir, K., Kılıc, Ö., & Gökmen, M. (2003). A Turkish handprint character recognition system. In Yazıcı, A. & Şener, C. (Eds.), *Computer and Information Sciences - ISCIS 2003*, (pp. 447–456)., Berlin, Heidelberg. Springer Berlin Heidelberg.

[11] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

[12] Conneau, A., Baevski, A., Collobert, R., Mohamed, A., & Auli, M. (2020). Unsupervised cross-lingual representation learning for speech recognition.

[13] Diaz, D. H., Qin, S., Ingle, R., Fujii, Y., & Bissacco, A. (2021). Rethinking text line recognition models.

[14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

[15] Dutta, K., Krishnan, P., Mathew, M., & Jawahar, C. (2018). Improving cnn-rnn hybrid networks for handwriting recognition. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (pp. 80–85).

[16] Fischer, A., Frinken, V., Fornés, A., & Bunke, H. (2011). Transcription alignment of latin manuscripts using hidden markov models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, (pp. 29–36)., New York, NY, USA. Association for Computing Machinery.

[17] Francis, W. N. & Kucera, H. (1979). Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

[18] Graves, A. (2012). Sequence transduction with recurrent neural networks.

[19] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009a). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 855–868.

[20] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009b). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 855–868.

[21] Graves, A. & Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. In Koller, D., Schuurmans, D., Bengio, Y., & Bottou, L. (Eds.), *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

[22] Grosicki, E., Carré, M., Brodin, J.-M., & Geoffrois, E. (2009). Results of the rimes evaluation campaign for handwritten mail processing. In *2009 10th International Conference on Document Analysis and Recognition*, (pp. 941–945).

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. (pp. 770–778).

[24] Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–80.

[25] Hu, J., Brown, M., & Turin, W. (1996). Hmm based online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *18*(10), 1039–1045.

[26] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, (pp. 448–456). JMLR.org.

[27] Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, (pp. 2017–2025)., Cambridge, MA, USA. MIT Press.

[28] Kahn, J., Lee, A., & Hannun, A. (2020). Self-training for end-to-end speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

[29] Kang, L., Riba, P., Rusiñol, M., Fornés, A., & Villegas, M. (2022). Pay attention to what you read: Non-recurrent handwritten text-line recognition. *Pattern Recognition*, *129*, 108766.

[30] Kang, L., Riba, P., Villegas, M., Fornés, A., & Rusiñol, M. (2021). Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture. *Pattern Recognition*, *112*, 107790.

[31] Kang, L., Toledo, J., Riba, P., Villegas, M., Fornés, A., & Rusiñol, M. (2019). Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition: 40th german conference, gcpr 2018, stuttgart, germany, october 9-12, 2018, proceedings. (pp. 459–472).

[32] Kass, D. & Vats, E. (2022). Attentionhtr: Handwritten text recognition based on attention encoder-decoder networks. In Uchida, S., Barney, E., & Eglin, V. (Eds.), *Document Analysis Systems*, (pp. 507–522)., Cham. Springer International Publishing.

[33] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (pp. 1097–1105)., Red Hook, NY, USA. Curran Associates Inc.

[34] Lee, C.-H. & Juang, B.-H. (1996). A survey on automatic speech recognition with an illustrative example on continuous speech recognition of Mandarin. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 1, Number 1, August 1996*, (pp. 1–36).

[35] Li, M., Lv, T., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2021). Trocr: Transformer-based optical character recognition with pre-trained models.

[36] Luo, C., Zhu, Y., Jin, L., & Wang, Y. (2020). Learn to augment: Joint data augmentation and network optimization for text recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13743–13752.

[37] Marti, U.-V. & Bunke, H. (2002). The iam-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, *5*(1), 39–46.

[38] Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models.

[39] Michael, J., Labahn, R., Gruning, T., & Zollner, J. (2019). Evaluating sequence-to-sequence models for handwritten text recognition. (pp. 1286–1293).

[40] Myers, E. W. (1986). Ano(nd) difference algorithm and its variations. *Algorithmica*, *1*(1), 251–266.

[41] Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (pp. 807–814)., Madison, WI, USA. Omnipress.

[42] Ney, H., Essen, U., & Kneser, R. (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech Language*, *8*(1), 1–38.

[43] Pechwitz, M. & Maergner, V. (2003). Hmm based approach for handwritten arabic word recognition using the ifn/enit - database. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, (pp. 890–894).

[44] Ploetz, T. & Fink, G. (2009). Markov models for offline handwriting recognition: A survey. *IJDAR*, *12*, 269–298.

[45] Poznanski, A. & Wolf, L. (2016). Cnn-n-gram for handwriting word recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 2305–2314).

[46] Puigcerver, J. (2017). Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, (pp. 67–72).

[47] Scheidl, H., Fiel, S., & Sablatnig, R. (2018). Word beam search: A connectionist temporal classification decoding algorithm. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, (pp. 253–258).

[48] Shanmugam, D., Blalock, D., Balakrishnan, G., & Guttag, J. (2021). Better aggregation in test-time augmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, (pp. 1194–1203).

[49] Shi, B., Bai, X., & Yao, C. (2017). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(11), 2298–2304.

[50] Shorten, C. & Khoshgoftaar, T. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*.

[51] Simard, P., Steinkraus, D., & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, (pp. 958–963).

[52] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

[53] Sánchez, J. A. (2016). Bentham dataset r0.

[54] Toiganbayeva, N., Kasem, M., Abdimanap, G., Bostanbekov, K., Abdallah, A., Alimova, A., & Nurseitov, D. (2021). Kohtd: Kazakh offline handwritten text dataset.

[55] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (pp. 6000–6010)., Red Hook, NY, USA. Curran Associates Inc.

[56] Vural, E., Erdogan, H., Oflazer, K., & Yanikoglu, B. (2004). An online handwriting recognition system for Turkish. In *Proceedings of the IEEE 12th Signal Processing and Communications Applications Conference, 2004.*, (pp. 607–610).

[57] Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

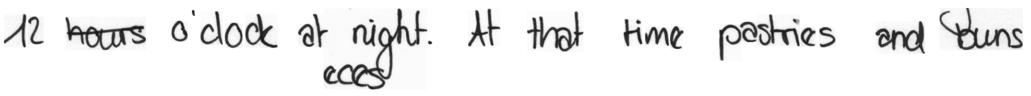
[58] Wigington, C., Stewart, S., Davis, B., Barrett, B., Price, B., & Cohen, S. (2017). Data augmentation for recognition of handwritten words and lines using a cnn-lstm network. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, (pp. 639–645).

[59] Xiao, S., Peng, L., Yan, R., & Wang, S. (2019). Deep network with pixel-level rectification and robust training for handwriting recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, (pp. 9–16).

[60] Yanikoglu, B. & Kholmatov, A. (2003). Turkish handwritten text recognition: A case of agglutinative languages. volume 5010, (pp. 227–233).

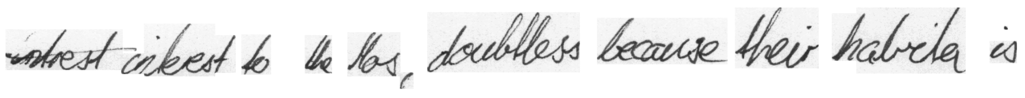
APPENDIX A
Incorrectly Labelled IAM Test Samples

Figure A.1 below illustrates samples from IAM test split with erroneous labels. The corrected labels of the IAM test split will be shared in [1].



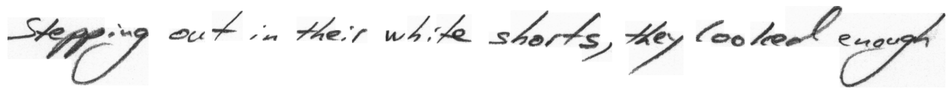
Label: 12 o'clock at night . At that time pastries and buns

Issue: Crossed-out 'hours'. another word under the line



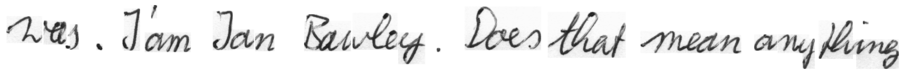
Label: interest to the Mos , doubtless because their habitat is

Issue: Crossed-out word before 'interest'



Label: Stepping out in their white shorts , they looked glad enough

Issue: 'glad' is not present

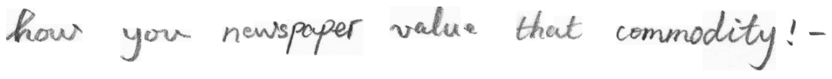


Label: was . I 'm Ian Bawley . Does that mean anything

Issue: Labelled as I'm instead of I'am

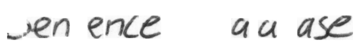
Label: Sentence Database P02-109

Issue: Completely wrong except the last a few letters



Label: know how you newspaper people value that commodity !-

Issue: 'know' is not present



Label: --

Issue: Completely wrong.

Figure A.1 Samples from IAM dataset test partition with wrong labels. *Label* denotes the ground-truth text written on the image and *Issue* indicates the error of the label e.g. a word in a label is not present in the corresponding image.

[1] https://github.com/firatkizilirmakk/handwriting-recognition