Sequence Learning for OCR in Unsupervised Training Cases

Thesis approved by
the Department of Computer Science
of the Technische Universität Kaiserslautern
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)

to

Martin Jenckel

Date of Defense:	24th February 2021
Dean:	Prof. Dr. Jens Schmitt
Reviewer:	Prof. Dr. Prof. h.c. Andreas Dengel
Reviewer:	Prof. Dr. Marcus Liwicki
Reviewer:	Prof. Dr. Faisal Shafait

D386

Abstract

Sequence learning describes the process of understanding the spatio-temporal relations in a sequence in order to classify it, label its elements or generate new sequences. Due to the prevalence of structured sequences in nature and everyday life, it has many practical applications including any language related processing task. One particular such task that has seen recent success using sequence learning techniques is the optical recognition of characters (OCR).

State-of-the-art sequence learning solutions for OCR achieve high performance through supervised training, which requires large amounts of transcribed training data. On the other hand, few solutions have been proposed on how to apply sequence learning in the absence of such data, which is especially common for hard to transcribe historical documents. Rather than solving the unsupervised training problem, research has focused on creating efficient methods for collecting training data through smart annotation tools or generating synthetic training data. These solutions come with various limitations and do not solve all of the related problems.

In this work, first the use of erroneous transcriptions for supervised sequence learning is introduced and it is described how this concept can be applied in unsupervised training scenarios by collecting or generating such transcriptions. The proposed OCR pipeline reduces the need of domain specific expertise to apply OCR, with the goal of making it more accessible. Furthermore, an approach for evaluating sequence learning OCR models in the absence of reference transcriptions is presented and its different properties compared to the standard method are discussed. In a second approach, unsupervised OCR is treated as an alignment problem between the latent features of the different language modalities. The outlined solution is to extract language properties from both the text and image domain through adversarial training and learn to align them by adding a cycle consistency constraint. The proposed approach has some strict limitations on the input data, but the results encourage future research into more widespread applications.

Acknowledgements

I am grateful to and want to thank everyone who supported me during my time as a PhD candidate which culminated in this thesis.

I want to especially thank Prof. Andreas Dengel for believing in my abilities and giving me the opportunity to work in the vibrant research enviroment at the Smart Data & Knowledge Services group at the DFKI. I also want to thank my supervisor Dr. Saqib Bukhari ,who guided me through the early stages of my research and always provided valuable feedback and advice, as well as Dr. Thomas Kieninger and Dr. Frederico Raue, both of which acted as stand-in supervisors during the final stages of my research.

I want to further extend my gratitude to all of my colleagues at the DFKI who created such a fun and dynamic working enviroment. I have to thank especially my long time office mates Hassan, Riaz and Gagan. I treasure the deep discussions, fun acitivities and many laughs we shared, which went well beyond the walls of the DFKI. You made the experience so much more fun and enjoyable.

A special thanks also goes to all of my friends outside of the DFKI. You guys kept me going through the difficult stretches and helped me get to the finish line. I also want to use this opportunity to thank Thomas Mesikapp and my dear sister for helping with proofreading this thesis on short notice.

Last but not least I want to thank my family, who supported me through all this time and never doubted my ability to achieve my goals. I would like to dedicate this achievement to my parents, my grandparents and my sister. I may have been far away from home and we only saw each other rarely, but the knowing support of you all has always been the most important to me. It is what enabled me to embark on this journey.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI Artificial Intelligence.

ANN Artificial Neural Network.

BILSTM Bidirectional LSTM.

CER Character Error Rate.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

GPU Graphics Processing Unit.

GRU Gated Recurrent Unit.

LPIPS Learned Perceptual Image Patch Similarity.

LSTM Long Short Term Memory.

RNN Recurrent Neural Network.

Seq2Seq Sequence-to-Sequence.

WER Word Error Rate.

CHAPTER 1

INTRODUCTION

Recent years have seen a surge in machine learning and artificial intelligence (AI), fueled by the increased computational power of modern hardware, especially CPUs and GPUs. Alongside more efficient computation techniques this allowed full use of the long established machine learning concept of artificial neural networks (ANN) for the first time (Krizhevsky et al., 2012). As a result machine learning techniques found many new applications in domains such as self-driving cars (Bojarski et al., 2016), traffic management (Lv et al., 2014) or health care (Ravi et al., 2016).

This also led to improvements in the long standing machine learning domains related to language processing. The use of recurrent neural networks (RNN), a special type of ANN that enables the learning of sequential structures, combined with a gated memory neuron model called Long Short Term Memory (LSTM) cell (Hochreiter and Schmidhuber, 1997), allowed for new and more robust AI applications. Through the direct modelling of the sequential properties inherent to language, like grammar and semantics, and applying them to its own prediction, AI has achieved human-like performance in multiple language processing tasks. In particular, the field of optical character recognition (OCR) has benefited from this development through a novel technique to create specialized AI for the recognition of text sequences (Graves, Liwicki, et al., 2008). This not only improved the quality of OCR, but also streamlined the application of modern machine learning techniques to text recognition tasks.

In the following Section 1.1 first the limitations to this novel approach are highlighted before presenting how some of these limitations will be addressed

Figure 1.1: Historical documents from the 15th century The three sample pages are from three printings of the same book. Although they have the same content, they are printed with different layouts, in different languages using different fonts, resulting in each requiring different expertise to produce a high level transcription. Provided by the Kallimachos project [1].

in this thesis. In Section 1.2 this is formalized into a research question and corresponding hypotheses. The contributions resulting from answering this question are listed in Section 1.3 and finally an overview of the thesis's structure is given in Section 1.4.

1.1 Motivation

The introduction of high performance OCR through LSTM-RNN based sequence learning has greatly advanced many fields that process sequential data, including the field of OCR, but in order to achieve this high level of performance the RNN has to be trained with large amounts of data. Similar to human learning, this training relies on external feedback for every good or bad decision made during the learning process. The most common approach involves presenting the AI with many separate problems and their corresponding solutions, so it can learn how to solve them and infer how to solve new problems. The high learning capacity of ANN in particular is mirrored by their need for many such pairs to properly learn their respective tasks. In OCR as well this introduced a new need for large amounts of transcribed

[1] http://kallimachos.de/kallimachos/index.php

reference data. For contemporary documents this usually does not pose a problem, since many documents are already generated digitally and collecting sufficient data is relatively easy.

At the same time there have been vast digitization efforts of archives and libraries [2], with the goal of conserving documents and making them accessible. As a result scanned document images for various historic scripts, languages and fonts are readily available and wait for robust OCR approaches to be fully digitized. For these documents manually collecting large training data sets is less feasible and comes at a high price in terms of manual labor to create the needed reference transcriptions. As a result high performance OCR engines are still not widely available for many historical documents, but those with the most common scripts and languages. Similar problems can also be found in related text recognition tasks such as handwriting recognition or recognising texts in natural scenes.

This thesis will address the issue of training sequence learning OCR models in the absence of reference transcription in three ways:

- It introduces the use of erroneous transcriptions as training data for LSTM-RNN based OCR models and presents how to collect or generate them,
- it describes a way to evaluate trained OCR models in the absence of reference data through the use of cycle consistency, and
- it outlines the use of adversarial training with input reconstruction for OCR, which allows for unsupervised training with mono-domain data.

1.2 Hypothesis

Given the introduced problem, the main research question this thesis tries to answer is

Question How can sequence learning be used to train state-of-the-art OCR models when no transcribed training data is available.

[2] https://www.primaresearch.org/projects/IMPACT

There are multiple possible answers and this thesis can not present an exhaustive answer. It will focus on concepts that make full use of the available image data, while mitigating the lack of corresponding transcriptions. The following three hypotheses are formulated in an attempt to answer this question:

H1 An LSTM-RNN can learn from and improve erroneous transcription data in sequence labeling tasks.

H2 Input reconstruction in the image domain can provide a measure for OCR performance.

H3 Adversarial training combined with cycle consistency enables unsupervised training for LSTM-RNN based OCR models.

1.3 Contributions

To answer these research questions and validate the hypotheses, this thesis has made several contributions to the training, architecture and evaluation of LSTM-RNN based OCR models. Practical contributions that support broader conceptual contributions are grouped accordingly. These contributions are:

C1 The training of LSTM-RNN based OCR with erroneous transcriptions. This enables the use of collected or synthetically generated low quality transcriptions and produces competitive results. (see Section 5.1)

C1.1 A processing pipeline for combined synthetic ground truth generation and LSTM-RNN model training. The proposed pipeline generates synthetic ground truth through clustering and minimal manual labeling and uses it as erroneous transcriptions for LSTM-RNN training. (see Section 5.2)

C2 The training of LSTM-RNN based OCR with uncertain transcriptions. Allowing the annotator to put uncertain annotations into the transcription helps closing the gap between expert and layman transcriptions, while providing the network with additional information. (see Section 5.3)

C2.1 A modified CTC-based LSTM-RNN OCR training algorithm to allow for uncertainty in the target sequence. The changes to the training objective enable the use of fuzzy annotations and increase the networks reliance on contextual information. (see Section 5.3.2)

C3 The transcription-free LSTM-RNN based OCR model evaluation through input reconstruction. The proposed approach links OCR errors to errors in the reconstruction of the input image and discusses the differences to the standard text based evaluation. (see Section 6)

C3.1 An encoder-decoder LSTM-RNN architecture and relative OCR model ranking scheme for transcription-free OCR model selection. In more detail, an LSTM-RNN is used as a decoder to generate text line images from OCR output. The trained decoder can be combined with different OCR models for an evaluation based on a relative model ranking. (see Section 6.1)

C4 A cycle consistent adversarial approach for unsupervised LSTM-RNN based OCR model training. Adding cycle consistency to adversarial training provides additional grounding of each symbol and enables the unsupervised approach. (see Section 7.2)

1.4 Thesis Overview

Following this introduction the thesis is structured into 8 chapter. Chapter 2 to 4 provide all the background information necessary and introduce various related and previous works. Chapter 2 further introduces the concepts of sequence learning. It gives definitions for the various sequence learning tasks and introduces the general approaches that are used to solve them. It also provides a better explanation on what unsupervised training cases are and highlights their importance. It is followed by Chapter 3 that focuses on introducing the details of the LSTM-RNN standard OCR model, how it can be trained and why it relies on transcription data. Chapter 3 finishes with an introduction to adversarial networks and discusses their advantages and disadvantages. The background information about OCR as an application domain of machine learning can be found in Chapter 4. Besides various related and previous works, it also gives an overview of the main application domain for the work presented in this thesis, historical documents.

The following three Chapter 5 to 7 describe the mentioned contributions and conducted experiments as well as discuss their results. Chapter 5 starts with the evaluation of LSTM-RNN training for OCR with erroneous transcriptions and provides a practical application pipeline combining clustering, limited annotation and supervised LSTM-RNN training into a semi-supervised OCR

approach. It also introduces the use of uncertain transcriptions called fuzzy ground truth, as a method to limit the gap between layman and expert annotations and as an alternative source of erroneous transcriptions. In Chapter 6 the focus lies on evaluating trained LSTM-RNN based OCR models in a transcription-free manner. A solution using cycle consistency in the image domain as an informative measure of OCR performance is described and its advantages as well as its disadvantages discussed. The last contribution regarding a cycle consistent adversarial approach for unsupervised OCR can be found in Chapter 7. Inspired by the architecture described in Chapter 7 it starts with evaluating various training schemes using unlabeled training data, before outlining a possible future for unsupervised OCR training through the adversarial approach.
Finally Chapter 8 concludes this thesis and gives a brief outlook towards possible future work.

CHAPTER **2**

SEQUENCE LEARNING

Sequences are a common part of every day life, such as the characters and words in language, the thought processes during reasoning or the bases in DNA. These sequences have in common, that their elements are not in a random order, but rather depend on each other to form a specific structure. In order to form a coherent sentence the characters and words have to be ordered according to the language's structure and grammar. Similarly running requires the leg muscles to contract and relax in a certain order to perform a fluid and efficient motion. Sequence learning aims to extract, recognize and use these structures to solve various classification, labeling or generation tasks.
With the common appearance of structured sequences it is not surprising that sequence learning has found applications in many domains, such as cognitive sciences (Kidd et al., 2018), financial engineering (Bao et al., 2017), DNA sequencing (Hoff et al., 2016), time series analysis (Zou et al., 2019), robotics (Inoue et al., 2019) and artificial intelligence (AI) (Graves, 2012). This work focuses on machine learning applications of sequence learning for optical character recognition (see Chapter 4).

This chapter provides the background and definitions for the tasks of sequence classification and labeling in Section 2.1 as well as the task of sequence generation in Section 2.2 from a machine learning perspective. In that context it also provides a definition of supervised and unsupervised training cases and discusses the related symbol grounding problem in Section 2.3.

2.1 Sequence Classification and Labeling

When trying to understand an unknown sequence there are two questions that arise:

- What type of sequence is it?
- What are the elements in the sequence?

Both of these questions have the goal of simplifying the process of understanding this unknown sequences. Knowing the type of sequence and being able to distinguish it from other sequences provides semantic and contextual information. It can help identifying what type of prior knowledge is applicable to the sequence. Being able to distinguish a question from a statement informs the listener that an answer is expected from him. With the additional knowledge about the words and their meaning, the listener can then understand it and formulate an answer.

Sequence classification is the task that tries to answer the first question. Its goal is to identify the group or class a given sequence is part of.

Definition 2.1.1 (Sequence Classification)
Given a sequence x, a set of labels $L = \{l_1, l_2, ..., l_k\}$ and a sequence classifier F, find the label $y = F(x); y \in L$.

The general strategy is to accumulate the information of the whole sequence into a single descriptor, which is then classified (Tolosana et al., 2018).

The task of assigning each element within a sequence to a group is called sequence labeling. Here the goal is to find the correct label for each sequence element.

Definition 2.1.2 (Sequence Labeling)
Given a sequence x with a segmentation $S(x) = x_1, x_2, ..., x_n$, a set of labels $L = \{l_1, l_2, ..., l_k\}$ and a function H, find the sequence $y = y_1, y_2, ..., y_n; y_i \in L$ such that $y_i = H(x_i|x)$

For a sequence with a single element, both sequence labeling and sequence classification are identical.
Some researchers further divide the task of sequence labeling into *segment labeling* and *temporal classification* based on whether an explicit segmentation

is used or not (Graves, 2012). Given the added difficulty for finding the segmentation, recent research has mostly gone in the direction of the latter (Breuel, 2015; Graves, Fernández, et al., 2005; Schmidhuber et al., 2002). These approaches make use of the restricted order of sequence elements to align the input and the target sequence.
Sometimes however it is necessary to find the segmentation explicitly:

Definition 2.1.3 (Sequence Segmentation)
Given a sequence x and a set of segments X, find a segmentation S such that $S(x) = x_1, x_2, ..., x_n; x_i \in X$.

In many application scenarios with real data, the segments x_i are noisy versions of the reference segments, requiring additional matching between noisy segment candidates and their counterparts in X. It is therefore common to make use of prior knowledge about the sequence structure, such as the space between words in written English text (Louloudis et al., 2009; Papavassiliou et al., 2010). In the absence of such knowledge possible segment candidates are generated and validated either through a classifiers ability to label a segment (Roy et al., 2009) or through corresponding segments in the reference data (Qin and Manduchi, 2017).

2.2 Sequence Generation

Due to the nature of assigning labels to each sequence segment in a sequence labeling task, the output y is also a sequence. Similarly, the focus of sequence generation is to generate new sequences, but without the need of an alignment between input and output sequence.

Definition 2.2.1 (Sequence Generation)
Given a target set Y over all target sequences, an input sequence $x = x_0, x_1, ..., x_n; x \in X$ and a generator G, generate a sequence $y = G(x) = y_1, y_2, ..., y_m$ such that $y \in Y$.

The difficulty of this task often depends on the similarity of the input space X and the output space Y.
Common applications of sequence generation include sequence translation (Conneau et al., 2017; Lample et al., 2017) and sequence prediction (Bahdanau, Brakel, et al., 2017; Srivastava et al., 2015). As the name implies the former's goal is to translate between input and output domain by generating

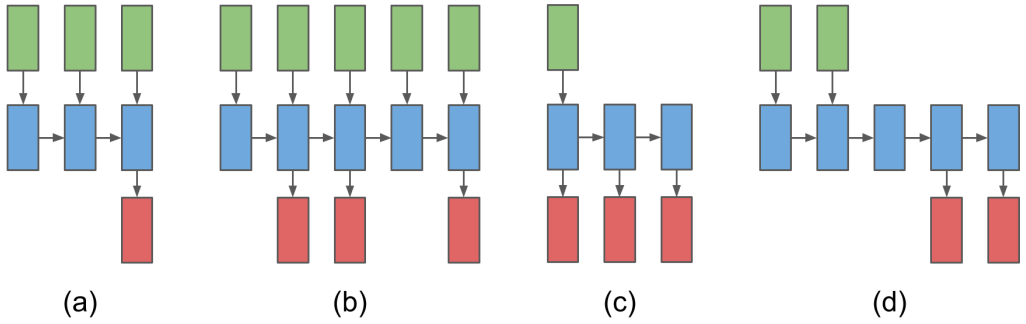

Figure 2.1: Flexibility of sequence learning Sequence learning needs flexibility to learn all the required mappings between the input (green) and output (red) sequence. These include many-to-one mappings (a) for sequence classification, many-to-many mappings (b) for sequence labeling, one-to-many mappings (c) for sequence generation or time-delayed mappings (d) for sequence prediction.

specific sequences $y \in Y$ from inputs $x \in X$. The latter is a special case, where the input space X and the output space X^c are complements of a target space $Y = X + X^c$. After the prediction of x^c, the combination of input sequence x and output sequence x^c then is a complete sequence $y = x + x^c \in Y$.
In recent years generative approaches have become more popular and have found widespread applications, from generating hand writing (Graves, 2014) or speech imitation (Oord, Dieleman, et al., 2016), to composing music scores (C.-Z. A. Huang et al., 2018). These applications have in common, that they ignore the conditional dependency between the input and target domain and instead focus on generating a random, but realistic sequence in the target space.

2.3 Supervised vs Unsupervised Training Cases

Even though the tasks of sequence classification, labeling and generation have been introduced with the goal of producing a desired output, machine learning is more interested in creating an autonomous agent that learns to produce these outputs. In sequence learning this involves learning the underlying structure of a sequence and translating that knowledge into a specific output.

A human can read a book or take a lecture to learn about a new topic and test his acquired knowledge by applying it to a problem. Through the comparison of his result with the real solution he can then receive feedback from his surrounding, e.g. a teacher or a text book. Similarly machine learning heavily relies on the availability of correct solutions for training problems. An artificial agent can make a prediction and correct its understanding of the problem based on how far off its prediction was from the correct solution. This type of training is called supervised learning, as the correct solution to the training problem serves as an external supervision of the learning process.

Definition 2.3.1 (Supervised Learning)
Given an input domain X, a target domain Y, a joint distribution $D_{X\times Y}$ and an error function $E : Y \to \mathbb{R}$, the agent has to learn a function $P : X \to Y$ from sample pairs $s_i = (x_i, y_i); s_i \in D_{X\times Y}$, such that the error $e = \sum_i E(P(x_i), y_i)$ is minimized.

Not all target domains provide the same level of supervision. A target like the pixel wise annotation of characters in a text image provides a stronger supervision, than only a sequence of Unicode labels for characters that appear somewhere in the same image. Supervised learning is the most common type of learning since the agent can infer the correlations between inputs and targets directly through the supervision provided by the problem-solution pairs.

If an approach can learn without the external supervision provided by input-target pairs, it is considered unsupervised learning.

Definition 2.3.2 (Unsupervised Learning)
Given an input domain X, a target domain Y and an error function $E : Y \to \mathbb{R}$, the agent has to learn a function $P : X \to Y$ from inputs $x_i; x_i \in X$, such that the error $e = \sum_i E(P(x_i))$ is minimized.

Without the correlation between inputs and targets, unsupervised learning relies on finding other constraints. These can be assumptions, like similar inputs belong to the same group (Aghabozorgi et al., 2015), or self-consistency in reversible processes (Cho et al., 2014). In recent years unsupervised learning has become a bigger research focus with the introduction of adversarial training (Goodfellow, Pouget-Abadie, et al., 2014). In work by Lample et al., 2017 adversarial training for Neural Machine Translation (NMT) has been introduced to avoid the need of input-target pairs for training. Instead the agent learns to directly align the sequential properties of the input and

the target domains through the use of unpaired training samples. Similar approaches also spread to related domains such as Speech-To-Text (Chung, Weng, Tong, and J. R. Glass, 2018) or OCR (Gupta et al., 2018a).

The two definitions given for supervised and unsupervised training do not cover all machine learning approaches. *Semi-Supervised Learning* describes training approaches that make use of both, paired input-target training samples as well as training data without input-target pairs. This allows semi-supervised learning to make use of the advantages of both supervised and unsupervised learning. A widespread example for this type of learning is unsupervised pre-training (Bengio et al., 2006). As the name suggests an agent first learns general data properties from unpaired training data in an unsupervised way. Then the pre-trained network is fine-tuned on paired training data with the assumption, that the pre-learned properties are useful descriptors for the supervised learning.
Another type of learning that can neither be grouped with supervised or unsupervised learning is *Reinforcement Learning* (RL). In RL the agent's goal is not to approximate a target output as close as possible, but rather to produce an output that maximizes a reward r. Given an environment G in a state g_i, the agent has to perform an action t_{ij} that interacts with the environment to receives a reward r_{ij} based on the chosen action and the new state of the environment g_j. Due to the sequential nature of repeated state transitions, RL can be treated as a form of sequence learning. It also tries to solve similar problems, such as finding the optimal sequence of actions when playing a game (Mnih et al., 2015). While there has been a surge in recent years towards RL, applications for RL in OCR related fields have had only minor success (Abtahi et al., 2015). This work therefore focuses on the other three types of learning approaches introduced in this Section.

Similar to the categorization of learning methods, a corresponding categorization for application scenarios can be defined according to the available data.

- *Supervised Training Cases* are application scenarios where training samples are available as input-target pairs.
- *Unsupervised Training Cases* are application scenarios where no paired training samples are available.

Making the distinction between unsupervised training and unsupervised training cases is important, because unsupervised training methods are not

the only way of approaching an unsupervised training case. While the well studied machine learning problems are supervised ones, unsupervised training cases are much more common. Incentivized by the highly developed state-of-the-art supervised training methods, unsupervised training cases are often made into supervised or semi-supervised ones. This can happen either via *transfer learning* using related available training data, generating *synthetic training data* or manually creating the required input-target samples. Possible are also combinations of these, such as using pre-trained models or *augmenting* existing data synthetically. In some scenarios the process of creating or collecting training data can be prohibitively difficult, time consuming or expensive, such that unsupervised learning methods provide a better alternative.

2.3.1 The Symbol Grounding Problem

One reason it is harder to use unsupervised training methods is the symbol grounding problem. It was introduced by Harnad, 1990 and deals with the question how an autonomous artificial agent can learn the relationship between its internally learned representations and a set of unique symbols given to itself. Hanard formulated it in the following way:

> "How can the meanings of the meaningless symbol tokens, manipulated solely on the basis of their (arbitrary) shapes, be grounded in anything but other meaningless symbols?" (Harnad, 1990)

The main problem is that the symbols given to an agent carry no intrinsic meaning. It only arises semantically through the association of the symbols with their representations.
Lets take for example the concept of the character A as a symbol. This symbol only has meaning to a human if he has grounded it by associating it with a written shape, a sound and so on. An agent who can independently learn the various shapes in written text and the structural properties of the underlying language, would still have a problem, not knowing which of those character shapes is an A. The original problem is of a more philosophical nature and both theoretical and practical solutions to it have been proposed (Steels, 2008; Taddeo and Floridi, 2005).
In computer science there is a bigger practical interest of this problem when dealing with unsupervised training approaches. Supervised learning approaches handle the symbol grounding problem indirectly through the input-target samples. This way the grounding of the symbols is introduced extrinsically, since for every sequence element, the agent is informed about the

corresponding symbol through the target output. The agent can therefore ground the target output symbols with the learned representations of the inputs through the correlation of the two.
If the agent is trained in an unsupervised manner, another extrinsic source of information such as human input is required. Another recurring theme is to incorporate prior knowledge about the desired distribution of symbols or their properties into the training scheme. The grounding of each symbol can then happen through matching these distributions and properties between the input and the target domain. Raue et al., 2018 constrained an agent by an explicit distribution while attempts by Lample et al., 2017, Chung, Weng, Tong, and J. R. Glass, 2018 and Gupta et al., 2018a use a secondary network to learn the distribution directly from the target domain.

In the following chapters this work explores various sequence learning approaches for OCR related unsupervised training cases. In Chapter 5 a semi-supervised training approach is proposed which addresses the grounding problem through a human interpreter, while the work outlined in Chapter 7 uses an unsupervised approach that directly matches latent language properties.

CHAPTER 3

RECURRENT NEURAL NETWORKS

In the first half of the 20th century bio-mathematicians have started to formalize the complex dynamics of neurons in the human nervous system with the goal of understanding and simulating the behaviour of biological neural networks (McCulloch and Pitts, 1943). The introduction of the Perceptron in 1958 (Rosenblatt, 1958) made these concepts applicable for early machine learning and AI tasks. The limited processing power available at the time slowed progress and artificial neural networks (NN) only became popular in machine learning and AI in 2012. That year Krizhevsky et al., 2012 showed that modern graphics cards can be used to efficiently train large neural architectures to a very high performance. Although the architecture of modern ANNs have become less similar to their biological inspiration, their problem solving power steadily increased to the point where a neural network powered AI rose to better-than human level play in the ancient Asian board game Go in 2015/16 (Silver et al., 2016), a game where predictions saw humans ahead of machines for at least another decade.

A special class of ANN are recurrent neural networks (RNN). They have recurrent connections that do not follow the strict layer to layer architecture of their feed-forward counterparts. These connections can connect neurons to previous layers or to themselves effectively forming loops in the network architecture. This adds a temporal dependency between neurons and layers in the network, on top of the spatial one . The recurrency makes these ANN especially efficient when dealing with the sequence learning tasks introduced in the last chapter. It allows the network to model sequential dependencies, which enables it to model sequential structures.

This chapter first introduces Long-Short-Term Memory as a special type of recurrent neuron architecture and highlights why it is necessary for efficient sequential learning processes. Then the theoretical foundations are laid for the different network architectures used in Chapter 5 to 7.

3.1 Long-Short-Term Memory

Recurrent connections allow for a lot of freedom in terms of network architecture and many different approaches such as Elman and Jordan networks (Elman, 1990; Jordan, 1997), Hopfield networks (Hopfield, 1982) or Echo state networks (Jaeger and Haas, 2004) have been proposed. Out of these, architectures with only self-recurrent connections are predominantly used in machine learning. This means that recurrent connections only exist between neurons of the same layer, keeping the layer architecture of the whole network independent from the temporal component of an RNN layer. A neuron with a self-recurrent connection can keep track of its output during the last time step, resulting in a short-term memory. While simple self-recurrency models are proficient at learning sequential data, their performance quickly drops the longer the time delay between cause and reaction in the data becomes. For many real world applications their memory is too short. Long-Short-Term Memory (LSTM) cells have been introduced by Hochreiter and Schmidhuber, 1997 to mitigate this problem, by designing an artificial neuron that, besides a self-recurrent connection, also has a cell state and a set of learnable gating mechanisms which allow the cell to keep its short-term memory indefinitely. The problem of RNNs forgetting their stored information after a few time steps is more commonly known as the *vanishing gradient problem*.
Before taking a more technical look into how LSTM solved this problem, the architecture of LSTM cells and the common way of training LSTM-RNNs for OCR are formalized briefly. A more detailed mathematical description of the LSTM and its standard training can be found in Graves, 2012.

3.1.1 Gated Memory Cells

LSTM cells have an additional cell state c_i, which acts as a memory unit, and three adjustable gates controlling the information flow through the cells as well as a set of peephole connections between these gates and the cell state. The input gate I controls how the input influences the cell state, while the output gate O regulates the cells output. The forget gate F was not part of

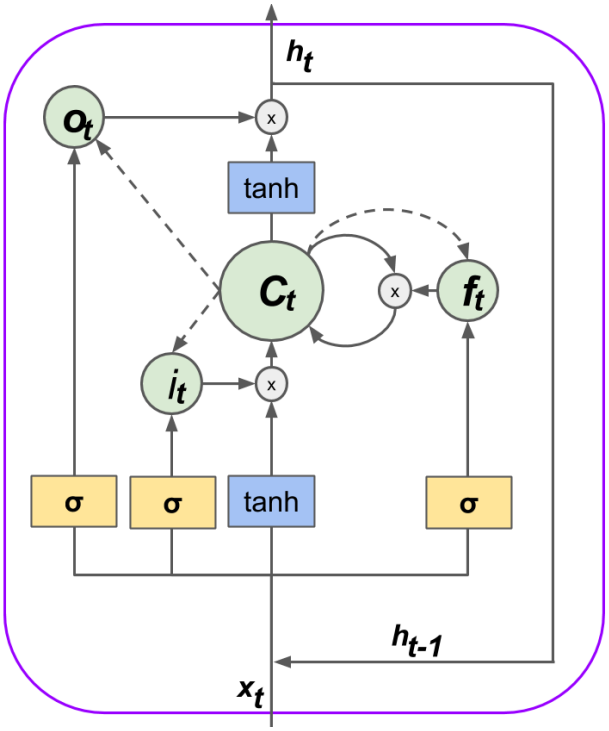

Figure 3.1: Schematic overview of an LSTM cell LSTM cells have a memory state C_t and several gates that regulate the information flow. The forget gate f_t and the peephole connections (dashed) were not part of the original architecture.

the original LSTM architecture but was later added by Gers, Schmidhuber, et al., 2000. It allows the cell to forget or recall a memorized state and adjusts the cell state's reliance on its memory. Similarly the peephole connections have been proposed later in Gers, Schraudolph, et al., 2002 to allow a cell's gates to adjust based on its own memory.
Since LSTM other gated memory cells like Gated Recurrent Unit (GRU) have been proposed (Cho et al., 2014), which are lightweight versions of LSTM cells that can process sequences faster. For many sequence learning tasks however, LSTM is still the state-of-the-art recurrent neural cell model.

Forward Pass

Feed-forward neurons have a set of inputs $x_1, ..., x_n$ and process them with a non-linear function σ to create a weighted and bias adjusted output activation.

The self-recurrency adds an additional set of inputs, which are the previous time step's outputs

$$a_h^t = \sigma(\sum_j w_{jh} x_j + \sum_k u_{kh} a_k^{t-1} + b_h) \tag{3.1}$$

For LSTM the forward pass includes additional computations for each gate and the cell state. To follow the information flow through the cell, the new output for each should be calculated in the following order:

Input Gate

$$i_h^t = \sigma(\sum_j w_{jh}^i x_j^t + \sum_k u_{kh}^i a_k^{t-1} + \sum_l v_{lh}^i c_l^{t-1} + b_i) \tag{3.2}$$

Forget Gate

$$f_h^t = \sigma(\sum_j w_{jh}^f x_j^t + \sum_k u_{kh}^f a_k^{t-1} + \sum_l v_{lh}^f c_l^{t-1} + b_f) \tag{3.3}$$

Cell State

$$c_h^t = f_h^t \cdot c_h^{t-1} + i_h^t \cdot \tanh(\sum_j w_{jh}^c x_j^t + \sum_l u_{lh}^c a_l^{t-1} + b_c) \tag{3.4}$$

Output Gate

$$o_h^t = \sigma(\sum_j w_{jh}^o x_j^t + \sum_k u_{kh}^o a_k^{t-1} + \sum_l v_{lh}^o c_l^t + b_o) \tag{3.5}$$

Cell Output

$$a_h^t = o_h^t \cdot \tanh(c_h^t) \tag{3.6}$$

The weights w, u and v with the upper indices i, f, o and c indicate the weighted connections for the inputs, the previous output activations and the previous cell state for the input gate, the forget gate, the output gate and the cell state respectively. σ denotes the non-linear Sigmoid function and $\tanh$ the hyperbolic tangent:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.7}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.8}$$

Backpropagation Through Time

The many free parameters of ANN, are trained through an approach called *backpropagation* (Goodfellow, Bengio, et al., 2016). Equivalent to how network inputs are propagated forward from neuron to neuron and layer to

layer, the idea is to propagate errors backwards through the network to every parameter. Based on the gradient of the error O with respect to a parameter p_i, which describes how the total error was influenced by the parameter, an update for each parameter can be calculated.

$$\partial_p O = \frac{\partial O}{\partial p_i} \tag{3.9}$$

Since the term "error" emphasizes that it should be minimized during training, the more general term *objective* is used.
RNNs can also be trained in the same way, but since the error is backpropagated through the time dimension rather than a spatial one, the process is called *backpropation through time* (BPTT) (Williams and Zipser, 1995). The main differences arise because the network parameters are shared between all time steps. The gradient with respect to the network parameters therefore has to be calculated for each time step, resulting in a cumulative gradient for each parameter.

$$\partial_p O = \sum_t \partial_p^t O \tag{3.10}$$

Backward Pass

Because the error is backpropagated backwards through the network when calculating the gradients, it is also knows as the *Backwards Pass* through the network. For a simple RNN the derivative of Equation 3.1 with respect to the input activation of the previous time step a_k^{t-1} is

$$\frac{\partial a_h^t}{\partial a_k^{t-1}} = \sigma' \cdot u_{kh} \delta_a^{t+1} \tag{3.11}$$

Here the first term σ' is the derivative of the Sigmoid function:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \tag{3.12}$$

For the LSTM cell there are two types of incoming gradients. Besides δ_a^{t+1}, the gradient coming from the next time step, there is also δ_c^{t+1}, the gradient saved by the cell state. With

$$\partial_{x_h}^t = \frac{\partial O}{\partial x_h^t} \tag{3.13}$$

the derivatives then take the form:

Cell Output

$$\delta_a^{t+1} = \sum_l u_{hk}^i \partial_{i_h}^{t+1} + \sum_l u_{hl}^f \partial_{f_h}^{t+1} + \sum_m u_{hm}^c \partial_{c_h}^{t+1} + \sum_n u_{hn}^o \partial_{o_h}^{t+1} \quad (3.14)$$

Output Gate

$$\partial_{o_h}^t = \tanh(c_h^t)\delta_a^{t+1} \quad (3.15)$$

Cell State

$$\delta_c^{t+1} = o_h^t(1 - \tanh(c_h^t)^2)\partial_{a_h} + f_h^{t+1}\delta_c^{t+1} + \sum_l u_{hl}^i \partial_{i_l}^{t+1} + \sum_k u_{hk}^f \partial_{f_k}^{t+1} + \sum_m u_{hm}^o \partial_{o_m}^t \quad (3.16)$$

$$\partial_{c_h}^t = i_h^t(1 - \tanh(c_h^t)^2)\delta_c^{t+1} \quad (3.17)$$

Forget Gate

$$\partial_{o_h}^t = c_h^{t-1}\delta_c^{t+1} \quad (3.18)$$

Input Gate

$$\partial_{o_h}^t = \tanh(\sum_j w_{jh}^c x_j^t + \sum_l u_{lh}^c a_l^{t-1} + b_c)\delta_c^{t+1} \quad (3.19)$$

The processing order again follows the information flow through the cell, this time backwards.

Vanishing Gradient Problem

Although the forward and backward equations for training a generic recurrent neural network can be nicely formalized, training it to learn long range sequential dependencies does not work very well. This is because the partial differential $\partial_{pi}^T O$ is solved through repeated applications of the chain rule. Each time step adds an additional factor $\frac{\partial a^t}{\partial a^{t-1}}$ to the gradient. From Equation 3.11 one can understand that a repeated multiplication with the network parameters and the Sigmoid derivative leads to a problem. With

$$\sigma'(x) \leq 0.25 \forall x \quad (3.20)$$

and networks weights

$$|u_{ij}| < 4 \quad (3.21)$$

the total gradient converges towards zero and the network becomes unable to learn. On the other hand, if the weights are big,

$$u_{ij} >> 1 \quad (3.22)$$

then the gradient can also explode and the network quickly diverges.
The LSTM architecture prevents this exponential decay through its separate internal memory state. The input, output and forget gate regulate the information and error flow through the cell and can trap an error in the memory. Mathematically this can be seen in Equation 3.16. Similar to the simple RNN the gradient flow from the output activations is scaled by the derivative of the Sigmoid function and the parameter weights, but the term $f_h^{t+1}\delta_c^{t+1}$ describing the gradient trapped in the memory depends only on the output of the forget gate f_h^{t+1}. In the original LSTM architecture this was a fixed value which guaranteed linear error flow through the cell, but even as a learnable parameter this architecture prevents the gradient from vanishing.

3.1.2 Connectionist Temporal Classification

In the previous chapter supervised training has been introduced as training with input-target pairs. For sequence classification this task is straight forward, however for sequence labeling the input and the target can both be sequences. Without information about the locations of the sequence elements, the network only has an order of the target labels, but no information on how they align with the input sequence.
Early solutions to this problem involved hybrid models e. g. with Hidden Markov Models (HMM) (Renals et al., 1994), where classification and alignment where solved separately. These approaches have been widely replaced with the introduction of Connectionist Temporal Classification (CTC) (Graves, 2012). Rather than requiring a hybrid architecture, CTC is a temporal classification loss that allows to directly train RNNs to label and align sequences as a single task. This is achieved by allowing the network to make classifications for a specific label at any time step in the sequence as long as the output sequence of labels aligns with the target sequence. CTC uses a Softmax layer that transforms the network outputs a^t into probability distributions $y^t = y^t_{l_0}, y^t_{l_1}, ..., y^t_{l_n}, y^t_{l_{blank}}$ over all labels $l_i \in L$, plus one additional "blank" label l_{blank}. The joint distribution over all time steps

$$p(z|x) = \prod_t y^t_z \tag{3.23}$$

therefore estimates the probability distribution $p(z|x)$ over all possible sequences $z = \{z_0, z_1, ..., z_T\}; z_i \in L \cup l_{blank}$. The goal of CTC is now to optimize this probability distribution estimate so it aligns with the real distribution of target sequences. In a more formal way CTC tries to maximize the likelihood that an output sequence z, generated for an input sequence x, aligns with the

target sequence $s = s_0, s_1, ..., s_N; s_i \in L; N \leq T$. The alignment between two sequences is defined such that, given a many-to-one mapping

$$M : L^T \cup l_{blank} \rightarrow L^N \tag{3.24}$$

s and z are aligned if $s = M(z)$. The mapping M first aggregates all repeated output labels that appear consecutively in the output sequence into a single label and removes all "blank" labels afterwards. For example, $M(aaa\_\_\_aaa\_\_\_bbb\_\_ccc) = M(a\_a\_b\_c) = aabc$ where _ denotes the "blank" label. Since all the output sequences $z \in M^{-1}$ that are mapped to a single target sequence s are mutually exclusive, the likelihood of a specific target sequence s given an input sequence x is the sum over the joint probabilities of the output sequences $z \in M^{-1}$.

$$p(s|x) = \sum_{z \in M^{-1}} p(z|x) \tag{3.25}$$

Forward-Backward-Algorithm

Calculating this large sum over all $z \in M^{-1}$ would be numerically unfeasible, if it wasn't for a dynamic programming based approach called *Forward-Backward-Algorithm*. It uses the fact that many of the possible output sequences z share sub-sequences $z_{0,k} = z_0, z_1, .., z_k; k \leq T$, so it recursively calculates their likelihood and stores the results for all sub-sequences starting from the first and last sequence element respectively. For simplicity the target sequences s' considered in the following equations are modified such that at the start and end as well as between each two consecutive label there is an additional "blank" label added, increasing the length of the sequence to $2N + 1$. The forward variable $\alpha^t(s)$ can then be defined as the sum over the distributions for all sub-sequences $z_{0,t}$, that get mapped by M onto a sub-sequence $s'_{0,t}$:

$$\alpha^t(n) = \sum_{z_{0,t}} \prod_t y^t_{z_{0,t}} \tag{3.26}$$

According to the properties of the mapping M and the way s' is constructed there are only three options for the label at z_t. First it can be a new label following a previous "blank" or it can be a "blank" following a previous "non-blank" label, given by $\alpha^{t-1}(n-1)$. Second it is possible that z_t is the same label as the previous output given by $\alpha^{t-1}(n)$. Finally, it can also be a new label s_n after the previous output z_{t-1} had the label s_{n-1}, given by $\alpha^{t-1}(n-2)$.

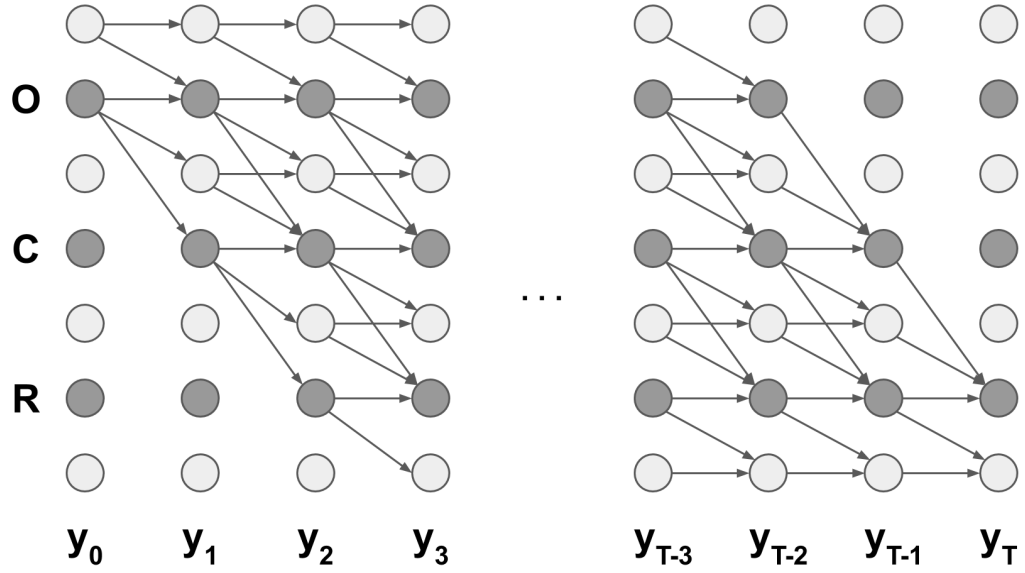

Figure 3.2: Graph visualization of the forward-backward algorithm The possible alignments between the target sequence (left) and the output distributions (bottom) can be calculated by the possible transitions (arrows) through the alignment graph. The forward variable for each state (sphere) represents the accumulated likelihood that the t-th network output represents the n-th target symbol. The bottom left and top right corner have no possible transitions due to the initial conditions, that each alignment has to start in the top left and end in the bottom right corner.

With this in mind the forward variables $\alpha^t(n)$ can be calculated recursively by:

$$\alpha^t(n) = y^t_{s'_n} \begin{cases} \sum_{i=n-1}^{s} \alpha^{t-1}(i) & \text{if } s'_n == \text{"}blank\text{"} \text{ or } s'_n == s'_{n-2} \\ \sum_{i=n-2}^{s} \alpha^{t-1}(i) & \text{else} \end{cases} \tag{3.27}$$

with the initial conditions:

$$\alpha^0(0) = y^0_{s'_0} \tag{3.28}$$

$$\alpha^0(1) = y^0_{s'_1} \tag{3.29}$$

$$\alpha^0(n) = 0, \forall n > 1 \tag{3.30}$$

The backward variables $\beta^t(n)$ can be defined in a similar way, but use the ending sub-sequence $z_{t,T}$ which is mapped to the ending target sub-sequence $s'_{l,N}$:

$$\beta^t(n) = \sum_{z_{t,T}} \prod_t y^t_{z_{t,T}} \tag{3.31}$$

It can be calculated recursively like the forward variable:

$$\beta^t(n) = y^t_{s'_n} \begin{cases} \sum_{i=n+1}^{n} \beta^{t-1}(i) & \text{if } s'_n == \text{"}blank\text{"} \text{ or } s'_n == s'_{n+2} \\ \sum_{i=n+2}^{n} \beta^{t-1}(i) & \text{else} \end{cases} \tag{3.32}$$

The initial conditions for the backward variables are also analogues:

$$\beta^T(n) = y^T_{s'_n} \tag{3.33}$$

$$\beta^T(n-1) = y^T_{s'_{n-1}} \tag{3.34}$$

$$\beta^T(n) = 0, \forall n < n-1 \tag{3.35}$$

By multiplying the forward and backward variables one gets

$$\alpha^t(n)\beta^t(n) = \sum_{z;z_t=s_n} \prod_t y^t_z \tag{3.36}$$

the likelihood for all output sequences with output $z_t = s_n$. Summing over all n for any t therefore gets the likelihood over all possible aligned sequences

$$p(s|x) = \sum_n \alpha^t(n)\beta^t(n) \tag{3.37}$$

Objective Function

The CTC objective has already been mentioned shortly in the context of Equation 3.25. Given an input sequence x, maximize the likelihood that output sequence z aligns with target sequence s. For a single training pair (x, s) the objective is given by:

$$O = -lnp(s|x) \tag{3.38}$$

To train a network with BPTT and apply the backward pass equations of the LSTM, one needs the gradient of the objective function with respect to the network outputs a^t_k.

$$\partial(a^t_k)O = y^t_k - \frac{1}{p(s,x)} \sum_{s_n=k} \alpha^t(n)\beta^t(n) \tag{3.39}$$

A detailed derivation of these gradients can be found in Graves, 2012.

Decoding

After an OCR model has been trained with CTC-loss and is applied to unseen data, it produces a sequence of character probabilities. That sequence has to be decoded to find the most probable sequence of characters $c \in L^R$

$$c = \underset{c}{\operatorname{argmax}}\, p(y|x) \tag{3.40}$$

of unknown length R. While there is no exact way to maximize this probability, there are multiple approximates that show good practical results (Graves, 2012). In this thesis a greedy decoding combined with the previously defined mapping M have proven to be sufficient.

3.1.3 Bidirectional LSTM

In spatial sequences there is no temporal order and the direction in which to provide the sequence as an input can be chosen freely. In those scenarios it can be beneficial to consider the sequential structure from both directions. This combines the accumulated sequential information from either direction for every output. In temporal sequences with defined causal relationships between sequence elements, it might seem counter intuitive to also process them in the reverse temporal order, but it can be beneficial as well in some cases. One every day example is humans inferring the meaning of unknown words in a sentence. Even though a reader would not start reading the sentence backwards, he would still consider how the sentence continues to get a better contextual understanding.
The kind of LSTM architecture, that processes a sequence in both temporal directions simultaneously is called a *bidirectional LSTM* (BILSTM) and consists of two LSTM-RNNs that work in parallel, one which processes the sequence in the normal direction and one that takes the reverse sequence as an input (Schuster and Paliwal, 1997). The forward LSTM accumulates information about the sequence up to a certain time step, while the reverse order LSTM accumulates information about how the sequence follows on after that time step. By combining the information the network is provided information about the whole sequence at every time step. Bidirectional architectures are common for many language related tasks where the full sequence is available at training time (Breuel, 2015; Graves, Fernández, et al., 2005; Tolosana et al., 2018).

3.2 Generative Adversarial Network

A *Generative Adversarial Network* (GAN), despite what its name suggests, does not describe a specific type of neural network, but rather a training method for ANN using an adversarial approach. It was introduced by Goodfellow, Pouget-Abadie, et al., 2014 as an unsupervised training method to learn the relationship between input and output domains, without the use of paired training data. Adversarial here indicates that two separate neural networks are pitted against each other in an attempt to outperform the other. The two networks are a discriminator, that distinguishes between real and fake samples, and a generator network, that in turn tries to generate fake samples to fool the discriminator. While there are not many restrictions on the architecture of either network, the training scheme dictates that the discriminator needs to output a fake-real probability, so it has a final fully connected layer with a single output value.
The generative part of the GAN is due to its original application domain, which is to generate random, but realistic sequences from a target domain, such as human faces. The determinism of the generator network, requires different inputs for different outputs. To simplify the sampling from the input space it is usually chosen to be $\mathcal{N}(0,1)^n$ or similar. The generator therefore learns to randomly map from a simple unit space to a highly complex target space. In the context of sequence learning this is especially useful for sequence generation tasks, where the focus lies on generating realistic sequences (Fedus et al., 2018; Yu et al., 2017).
In this thesis GANs are not used directly, but serve as an important foundation for an adversarial approach introduced in Chapter 7. This section will therefore focus only on the adversarial training of GANs. Further details on GANs in general and their use for generative tasks can be found in Goodfellow, Pouget-Abadie, et al., 2014.

3.2.1 Adversarial Training

The training for GANs is a two step process that trains the discriminator and generator networks respectively. For the discriminator training a real sample from the training data and a fake sample from the generator are evaluated and classified. During the discriminator training the target is to correctly classify the real sample as real and the fake sample as fake. Since the generator has the opposite objective, its parameters are kept fixed during discriminator training. The generator is trained similarly by creating a new

fake sample and passing it to the discriminator for evaluation. This time the goal is to fool the discriminator into classifying the fake sample as real. Here as well the discriminator's parameters are kept fixed during the generator training. In both training steps the classification errors are backpropagated through the networks. With the generator input x and a real sample s, the combined objective of the adversarial training can be formalized as

$$\min_{D}\max_{G} O(D,G) = \mathbb{E}_s[log(D(s))] + \mathbb{E}_x[1 - log(D(G(x))] \tag{3.41}$$

D and G here denote the discriminator and generator respectively.
With this adversarial training, the generator's ability to produce real-like samples relies on the discriminator's ability to classify them correctly. Only a negative feedback will incentivize the generator to produce better samples. The discriminator's ability to distinguish fake from real samples on the other hand depends on the available training data to sufficiently represent the target space as well as the quality of the fake samples.

Applying adversarial training comes with three practical problems. First the discriminator should not outperform the generator too much, because it effectively stops training for the latter. This is mainly a problem of the original GAN objective and can be mitigated by balancing the training between both networks or through an improved objective that is less sensitive to this (Gulrajani et al., 2017; Jolicoeur-Martineau, 2018; Mao et al., 2017). The second problem describes the exact opposite situation where the generator learns to generate samples only from a small sub-space, but successfully fools the discriminator. This *mode collapse* can be prevented by training more, so it is fooled less easily (Arjovsky et al., 2017). Lastly, if the training is too greedy, then the networks can also end up in a state where they periodically revisit the same set of solutions and never converge. More practical advice on GAN training can be found in Goodfellow, 2016.

CHAPTER 4

OPTICAL CHARACTER RECOGNITION

As humans we encounter sequential data on a daily basis, especially in the form of language. Either as text, speech or in our thoughts, language consists of characters from an alphabet, forming words and sentences. But not every combination of characters forms eligible words and only following the correct grammar creates understandable sentences. Predefined character sequences, such as words or phrases, and their semantic relationship defined by the grammar, encode information that a listener or reader can decode. The high amount of structure in these sequences is what allows for efficient communication through them.
For an artificial agent to perform the same task involves multiple complex processing steps, like digitization, language recognition and language processing. Different fields in computer science focus on automating these processes, depending on which medium is used for transmitting the message. Recognizing language in the form of written or printed text is called *Optical Character Recognition* (OCR).
This chapter introduces OCR as part of the broader field of *Document Analysis* and details the full processing pipeline required to apply OCR in Section 4.1. It also details different OCR approaches in Section 4.2 and introduces the special challenges when applying OCR to *Historical Documents* in Section 4.4.

4.1 Document Analysis

Since the invention of computers one of their main applications has been the automated processing of documents and every year the number of generated documents reaches record levels. Document Analysis aims at making the information in these documents accessible through automated processing. This includes indexing, extracting information, analyzing and transforming them. Automating these processes does not only save time, but also can enhance document understanding through additional meta-information or better accessibility.

But what are *Documents*? There are many different definitions of what exactly a document is, but they generally agree on their purpose. Documents are records that store information for the purpose of proving or reconstructing something (Buckland, 1997). Definitions by popular dictionaries on the other hand, focus on the most well known types of documents, like printed or written, but might not mention other domains of perception, like audio or video documents [1]. Documentalists, like Suzanne Briet, take a more theoretical approach to define documents. It is based on their function, rather than their physical representation (Buckland, 1997). According to her, even a sick animal, caught for the purpose of being studied, becomes a document of its sickness.
Although all work presented in this thesis only handles written and printed text documents, there is no need to limit the definition used here. Instead it is important to highlight, that there is a wide variety of other document types and corresponding avenues of document processing, that can serve as an inspiration for new processing techniques in OCR.

Historically the most important domain of document analysis was the recognition of text in image data. First commercial OCR systems started to appear from the late 50s into the 60s, at a time where almost all documents were generated physically and had to be digitized for automated processing or storage. Since then more and more documents are generated digitally and the focus of OCR shifted towards more specialized applications, such as handwritten documents (Graves, Liwicki, et al., 2008; Graves and Schmidhuber, 2009), complex languages (Ahmad et al., 2015; Ul-Hasan, Ahmed, et al., 2013) or historical documents (Breuel et al., 2013; Springmann et al., 2014). Similarly

[1] dictionary.cambridge.org

document analysis as a whole has expanded from OCR and other language related fields have been established.

One such increasingly important field is *Speech Recognition*. Instead of recognizing language through the visual domain, the goal of speech recognition is to processes audio recordings and recognize spoken language. While the input modality poses new challenges compared to OCR, audio and visual representations of language still share many underlying features, like word frequencies or sentence structure, and both domains can sometimes adapt similar approaches (Graves, Mohamed, et al., 2013).

Besides the task of transforming text and language between modalities, another common task with wide applications in document analysis is translating between different languages. The document analysis field *Neural Machine Translation* (NMT) aims at automating this process through the uses of ANN. Translating between languages has to solve different challenges compared to transforming between modalities.
When transforming language between modalities, even though its representation changes, the language's properties do not. This includes the sequential order of sequence elements and statistical features. Good translations however rarely allow for word-by-word translation and instead try to conserve the contained information. This requires the agent to understand the sequential nature of both the origin and the target language, in order to learn a sequential mapping that allows to rearrange, add and remove sequence elements Due to their differences NMT techniques are often not directly applicable to OCR and some adaptations are necessary (Bahdanau, Cho, et al., 2014; Lee and Osindero, 2016).

A third related field and the backbone of document analysis is *Natural Language Processing* (NLP). In the previously described research fields learning the language's underlying structure and rules is not the goal but the means to achieve certain tasks and as such learning often happens implicitly. NLP instead focuses on analyzing the syntactical structure and semantic relationships within a language explicitly. The specific task often depends on the language in question as well as the desired use for the extracted knowledge. Common syntactic NLP processors are stemming, part-of-speech-tagging, parsing and grammar induction, while semantic NLP tasks include extracting lexical or distributional semantics, language understanding, word disambiguation and named entity recognition.

NLP can be performed independently from other document analysis tasks, but is often used in combination. This way the extracted language properties can be used to further improve the various language processing tasks. For example, using the extracted knowledge as an additional restriction for the output sequences has been proven to be an effective way to improve OCR performance (Graves, 2012).

Between the various fields of document analysis there are many different approaches on how to solve the high variety of challenges. They can be defined as either *online*, where recognition of sequence elements happens live at the time of recording them, or *offline*, where the sequence is available completely at the time of recognition. Despite their differences all of the mentioned domains have in common, that they process language as sequential data and as such recurrent neural networks are a popular and effective tool. Similarly, due the approaches' or the data's various limitations, often pre-processing is required or beneficial, resulting in complex processing pipelines.
For OCR this pipeline usually consists of the four steps *Image Processing*, *Layout Analysis*, *OCR* and *Post-Processing* (Doermann, Tombre, et al., 2014).

4.1.1 Image Processing

In order for a computer to automatically process text-based documents, the document itself has to be digitized into an electronic image format via a scanner or a camera. The resulting image document often does not have the required image properties to be processed directly. Additionally the scanning process or the physical document might have a low quality, which would lead to a worse performance for most OCR systems.
Image processing therefore has two goals. The first is to adjust the image properties according to the requirements of the following processing steps, e.g. transforming a grey scale or color image into a binary one. The second goal is to mitigate the effect of bad image quality. This often goes hand in hand with the first goal.
The most common and often required image processing technique applied before OCR is *Binarization*. For many processing steps it is easier to process a document if they can focus only on certain foreground areas. As such Binarization has the goal of separating foreground and background pixel by labeling them with 0 and 1 respectively.
Another common effect of low quality scanning or physical documents are unwanted artifacts in the digitized image, such as noise and border effects.

denoising and *Despeckling* are two image processing techniques that identify and remove various types of noise. The additional step of *Cropping* can also remove unwanted artifacts outside of the document area by recognizing its border.
A second group of methods is used to straighten the image and text lines. *Deskewing*, as the name implies, aims at removing any skew from the image, often with respect to the contained text lines and not the original image orientation. Similarly, *Dewarping* reduces the effect of document warping like bends, riffles or similar distortions, often also with a focus on straightening the text lines. Both have the goal of making the extracting of structural elements and individual text lines easier during the next processing step.
It is noteworthy that for RNN-based OCR systems some image processing steps are less important and may even reduce performance. ANN in general are able to learn their own discriminative features and may perform better, if the information present in the raw image data is not artificially reduced. This applies especially to binarization and other processors, which reduce the information density in the image.

4.1.2 Layout Analysis

Real world documents have a huge quantity of possible layouts of text blocks, images, graphs, decorations, page numbers and so on. The OCR system on the other hand needs to know, where exactly the text elements are located, in order to recognize them. The main objective of layout analysis is to detect and extract text elements like lines, words or characters and to store useful meta-information about the layout. These include which language to recognize or whether a specific text region contains mathematical expressions or is a poem.
The analysis itself happens in a hierarchical manner, starting with the localization and classification of different text and non-text regions in the image, known as *block segmentation* or *logical layout analysis* depending on whether any linguistic features are used in the process. Then the page elements containing text are further separated into text lines as part of the *text line detection*. These text lines can then be further segmented into single characters if necessary, in a step called *text line segmentation*. This step is sometimes also called character segmentation (Casey and Lecolinet, 1996), but it describes the same process of segmenting a text line into single characters.
State-of-the-art RNN based OCR approaches like LSTM-RNN based OCR systems can perform the recognition directly on the text lines, which allows skipping the last segmentation step. This is especially important for hard to

segment texts like cursive scripts or languages with connected characters like Hindi or Arabic (Ahmad et al., 2015; Ul-Hasan and Breuel, 2013).

4.1.3 OCR

All the pre-processing done so far in the pipeline prepared the data for the recognition process. Whether the recognition happens character by character or on a full text line at once, the main steps for all OCR approaches are the same.
The first step is to identify features that allow the system to differentiate between the character classes. Especially for older OCR approaches the quality of the recognition highly depended on the quality of this *feature extraction*. Over time ANN-based approaches replaced most traditional ones, because of their ability to self-learn more descriptive features and outperform hand-crafted ones. This also meant creating an ANN-based OCR system no longer required the expertise in crafting these features and created a more accessible end-to-end training pipeline.
Based on these extracted features a classifier then makes decisions on what character is represented by each input. This *classification* either happens character by character or as part of the sequence labeling. During the latter, the classifier input is not only the extracted features, but also contextual information about the rest of the sequence.
The last OCR step involves combining the classifications results and finding the most likely character sequence given the output sequence of character probabilities predicted by the classifier. While there is not always a comprehensive way to do this *decoding*, there usually exist good practical methods to approximate it (Graves, 2012). In this thesis a greedy decoding is used, combined with the map introduced in Equation 3.24.
A detailed look into different OCR approaches can be found in the next section.

4.1.4 Post-Processing

Even though OCR systems can achieve quite high performance they rarely achieve 100% accuracy, even less so for more difficult application scenarios. Manual transcriptions do not always achieve this level either, so in either case it is worthwhile to apply some form of post-correction or quality check to the transcribed text.

This can be done with a *language model* that has learned the statistical properties of the underlying language and can identify the most probable word sequence based on the OCR output characters and local context. This is very similar to the sequential processing found in RNN-based OCR approaches, where these statistical properties are learned indirectly. In practice RNNs focus on the local statistics for character tuples and triplets and ignore most longer range dependencies, while explicit language models can be trained to put higher emphasis on these.
Another post-processing step is to enrich the OCR output with semantic and structural information, like the textual relationship between different text elements (Doermann, Tombre, et al., 2014). Humans extract this kind of information indirectly as they read a document and processes its layout. It highlights the importance of conserving the extracted layout information and re-applying it to the OCRed text output.
In the context of OCR, there is always the discussion between applying post-correction or using that additional information already during the OCR itself (Graves, 2012). If the OCR output is already improved, there might be no need for post-processing. Separate post-processing on the other hand is more flexible and can be applied if the OCR model itself is not accessible.

Applying OCR to a real document usually involves many of these processing steps, not just the OCR approach itself and the solution to any OCR-related problem can be severely influenced by every one of them. In this thesis the focus is on the OCR engine itself. All pre- and post-processing steps will be applied only when necessary and will be kept fixed where possible.

4.2 Related Work

Early OCR systems recognized characters independently from one another, heavily relying on their optical shapes as discriminative features. The languages sequential properties were only used in the decoding or post-pressing of the output. Due to their need of a character level segmentation, they will be called *segmentation-based* approaches in the following. This additional processing step introduces an additional error source, which is why they have mostly been replaced.
With RNN, HMM and CTC, tools became available that allowed processing of whole text sequences at once, without the need to segment them into

characters first. From here on these approaches will be called *segmentation-free* approaches. They rely on sequential processing which better models the structure of language, but also comes at a higher computational cost.
In the following, state-of-the-art approaches for both categories are introduced and their advantages and disadvantages are discussed for the use in unsupervised training scenarios.

4.2.1 Segmentation-based Approaches

Approaches in this category, as the name suggests, segment the input text into individual characters before recognizing them. This means the recognizer has to identify only one character per input, without the need for any alignment with the output sequence. A common approach is to make use of well established image classifiers, such as Support Vector Machines (SVM) or CNN, combined with either handcrafted (Belongie et al., 2002) or self-learned features (LeCun et al., 1998). Due to their lack of generality and the increasing performance of ANN, the former has mostly been replaced by the latter.
Similar to how ANNs combine feature learning and classification into a single training task, recent efforts have been made towards combining OCR with layout analysis into a single end-to-end trainable architecture that directly detects and recognizes characters from document pages. Clanuwat et al., 2019 introduced a purely convolutional model that jointly learns to detect character positions and classify them. This works especially well with complex layouts, where characters do not follow straight lines. However, the model also needs the exact character positions as training information, which poses a strong additional requirement on the annotation.
These approaches have in common, that they are supervised and require transcribed and annotated training data. The recognition of individual characters also allows the use of unsupervised machine learning techniques, such as clustering. Ho and Nagy, 2000; G. Huang et al., 2007; Junaidi, Fink, et al., 2011; Vajda et al., 2015 all have used clustering to show how to perform segmentation based OCR in an unsupervised or semi-supervised manner. Their approaches use clustering to reduce the complexity of the recognition to a single representative per group, which is feasible to be performed manually. Fully unsupervised OCR approaches have been achieved by using character frequencies and co-occurrence (Ho and Nagy, 2000) or n-gram language models for the final recognition (G. Huang et al., 2007) of cluster representatives.

Segmentation-based approaches have seen success with various OCR tasks, with some of them achieving very good performance within their applications domains. However their reliance on segmentation introduces an additional error source to the OCR system. Segmentation errors in particular are hard to compensate during the classification without the use of sequence learning and directly translate into recognition errors. Additionally, most high performance segmentation algorithms use supervised training, as they either require character level annotations with character bounding boxes or a pre-trained classifier. In the latter scenario the classifier is used to evaluate segmentation candidates through a dynamic programming approach (Smith, 2007). Unsupervised segmentation algorithms instead rely on language or script specific features, such as disconnected characters or fixed size character shapes, which makes it difficult to apply them widely.
In conclusion, segmentation-based approaches offer some options for unsupervised training cases, but they also introduce a new processing step and error source.

4.2.2 Segmentation-free Approaches

Handwriting and other cursive scripts as well as languages such as Hindi or Arabic, connect characters with one another or combine them into ligatures. Instead of better segmentation methods, research drifted towards finding classification methods that can process the connected text directly.
The most common form of segmentation-free approach and state-of-the-art for many application scenarios combines RNNs with a CTC-loss (Breuel et al., 2013; Graves, 2012). In this setup the RNN combines the shape features of its current input and combines them with the stored sequential information from previous and following inputs into a single input for the classifier. The classifier itself is a single ANN layer with Softmax activations, that predict a sequence of character probability distributions. The speciality is in how the CTC loss allows for the training with unsegmented text lines, by calculating errors based on all possible alignments between the output and target sequence.
Due to its popularity various improvements have been proposed. Breuel, 2017 proposed adding CNN layers prior to the RNN as a dedicated feature extractor to the architecture. A different change was proposed by (Cho et al., 2014), who replaced the original LSTM cells in the architecture with Gated Recurrent Units (GRU). Their reduced complexity lowers the architectures learning capacity only slightly while allowing for larger networks to be trained, which can lead to an improved overall performance.

A different approach to segmentation-free OCR is based on *attention*, a mechanism inspired by the human recognition of text. Attention models were introduced in NMT (Bahdanau, Cho, et al., 2014) first and required some adaptation before they found applications in OCR. For this Lee and Osindero, 2016 extended the concept from temporal to spatial attention. A spatial encoder, like a CNN, extracts high level features from the whole image and the attention model then lets the sequential decoder learn to focus on different regions of the input image during different steps in the decoding process. As a result the model learns the location and reading order of any text in the input image although it was never provided any information other than the target output text for each image.
For segmentation-free approaches as well there has been recent work on creating end-to-end processing pipelines that combine the OCR with, or allow to skip, certain pre-processing steps. Notably Tensmeyer and Wigington, 2019 introduced a system that works on whole document pages rather than text lines. This is achieved by formulating the detection of text lines and their alignment with the transcription as an additional optimization problem, that can be solved simultaneous to the OCR objective.

All three approaches, RNN plus CTC loss, RNN with attention and end-to-end learning, are supervised training approaches and would require a lot of manual transcription to apply them in unsupervised training cases. In general little work has been done with regards to segmentation-free unsupervised OCR. One exception is an early generative approach to OCR by Berg-Kirkpatrick et al., 2013. Unlike all other mentioned approaches, they solve the problem of OCR through the training of a complex generative model that learns to mimic the document image as close as possible by sampling characters from a language model. The correct text output for any image can then be identified as the input sequence, that leads to the best estimate of the output image. The downside of such an approach is the massive computational cost associated with the inference process, which can only partially be mitigated by heavy pruning.
Another more recent generative approach is by Gupta et al., 2018b, who trained an CNN-RNN architecture with unpaired training data. Their approach is heavily inspired by GANs. Given a text image, a generator produces a text output trying to fool a discriminator. The discriminator in turn tries to distinguish between real text from the training data and those produced by the generator. Although training requires data from both the image and the text domain, both the generator and the discriminator only need input from one domain each. No paired input-target samples are needed. Rather than matching image and text pairs, the system learns to match the language

properties between the two domains directly (Sutskever, Jozefowicz, et al., 2015).
In conclusion, segmentation-free approaches offer very good performance and high robustness, but are computationally expensive and mostly require large amounts of transcribed training data.

4.3 Evaluation

After performing OCR the question, how correct the OCRed text is, usually remains. Similarly, when creating an OCR system, the problem is to predict how this system will perform in an application scenario.
Besides human inspection of the results, there are two common evaluation metrics for OCR that are based on comparing OCRed text with a reference transcription and counting the number of errors. The *Character Error Rate* (CER) gives an estimate of how well the OCRed text matches the reference on the character level. It is usually calculated as a normalized Levenshtein-Distance (Levenshtein, 1966) between the OCR output text string and the reference ground truth text string. In particular it counts three different types of errors:

1 **Substitution Errors** are string mismatches between the output character and the corresponding ground truth character

2 **Insertion Errors** are string mismatches caused by the insertion of an additional character in the OCR output

3 **Deletion Errors** are string mismatch caused by a missing character in the OCR output

The total error describes the minimum number of such operations necessary to transform one of the strings into the other. It is normalized with the total number of characters in the reference string.
A more strict variant of the CER is the Word Error Rate (WER). Instead of the character level it counts mismatches between strings on the word level, counting the whole word as wrong if a single character mismatches. Although it is stricter than the CER, it also puts a much higher weight on the space character between words and has a slight bias towards longer words. Both variants are commonly used to report OCR results.

4.4 Historical Documents

OCR research has been and still is mostly fueled by its applications in the business world, administration and general everyday use. Its performance on modern printed text is therefore already very high, with accuracy in the range of $99\%+$. For the same reason research is focused on supervised OCR approaches, since training data for modern printed documents was readily available. Other open OCR problems, on the other hand, like robust handwriting recognition, minor languages like Arabian or Hindi, or historical documents have been of less interest. However, in the last decade more and more libraries and archives have started digitizing their inventories for accessibility, conservation and research purposes and the demand for OCR on historical documents has significantly increased.

4.4.1 Challenges

In the context of OCR, *historical document* is a loosely defined term for "old" handwritten or printed documents without any specified age limitation. However they have in common, that they pose additional challenges to the OCR pipeline.
When considering the optical recognition of text, the main challenge is a reduced image quality compared to contemporary documents. This is mainly due to the documents' age. The older the document the more likely it was printed or written on poor quality paper or to have sustained some damages or other degradations. Similarly printing and writing techniques have evolved over the centuries and older documents are more likely to contain unwanted artifacts, such as ink spillage or printing errors. Most of these problems lower the image quality and have to be addressed by improved pre-processing methods, but the OCR engine itself also can be improved to be more robust towards them.
Another challenge arises from the perspective of modern machine learning. Again the main reason is the documents age, but the relationship is more indirect. Training an RNN-based machine learning system requires a training data set that is large enough to reliably sample the whole underlying feature space. More specifically there needs to be enough training data with the same language and optical appearance. On the other hand, since printing and writing documents were not standardized, even within only a 50 year

period in a single region thousands of different fonts were used [2]. Similarly languages also had significant regional differences. This makes it difficult to assemble large coherent training data sets.
The size of the required training data is also a challenge with regards to supervised training. Transcribing damaged or otherwise degraded documents, containing varied fonts and nowadays no longer used or significantly changed languages, requires a high level of expertise and is time consuming. This can make supervised training prohibitively expensive and require different training approaches.

4.4.2 OCR Training

Similar to contemporary documents, LSTM-RNN based OCR models have become the standard for applying OCR to historical documents as well (Breuel et al., 2013; Moysset and Messina, 2019; Simistira et al., 2015; Yousefi, Soheili, Breuel, Kabir, et al., 2015). A lack of large transcribed training data sets has so far been addressed through Transfer Learning and Synthetic Training Data, two strategies that have been briefly introduced in Section 2.3. Both have been applied successfully to historical documents, but some challenges still remain.

Transfer learning has been applied to historical documents that share the same language, but more commonly to the same font as it guarantees a high level of optical similarity. In Reul et al., 2017 an LSTM OCR model was first trained as a mixed model with a collection of Latin, German and Dutch data sets with Fraktur and Antiqua font. The knowledge was then transferred to specialized models for each single data set via a fine-tuning step of only 60 or 150 text lines. In a second experiment, two available pre-trained models for modern English and generic Fraktur were used instead of the mixed model. In both experiments transfer learning managed to improve the OCR performance by up to 33%. However, the differences between the languages and their respective alphabets posed a problem, such that the properties and shapes of some characters had to be learned from scratch during the short fine-tuning phase.

[2] https://tw.staatsbibliothek-berlin.de/

When no such data is available, the alternative is to generate synthetic training data. A notable application of this can be found in Breuel et al., 2013, where a state-of-the-art OCR model for historic Fraktur has been trained using mostly artificially generated text line images. The resulting OCR model achieved a very high performance on historical documents and became the standard model for Fraktur in the OCRopus OCR system. This was mainly possible due to the existence of a good Fraktur font synthesizer that enabled the creation of high quality synthetic data. In general the performance of training with synthetic documents highly depends on the quality of the synthetic data and its ability to mimic the optical features of the real documents. Creating a good data synthesizer can be very time consuming and is also very reliant on a high level of domain specific expertise.

4.4.3 Narrenschiff

Due to the described challenges, historical documents mostly pose unsupervised training scenarios and are therefore a suitable application domain for this thesis. A 15th century Latin version of the novel "*Narrenschiff*" has been used for most of the experiments described in this thesis, which was part of the Kallimachos project [3].

The "Narrenschiff" is a satirical novel by Sebastian Brant and published by Johann Bergmann von Olpe in Basel from 1494 to 1500. In this time at least 28 different versions were printed, first in German and from 1497 also in Latin. These served as a basis for multiple unauthorized copies as well as later translations into English, Dutch, French and other languages. The data set used in this thesis consists of 102 scanned pages of an early Latin version and its corresponding line level transcription. All selected pages contain multiple text lines and a mostly consistent one column layout with additional side notes. Besides text, some pages also contain illustrations and other decorative elements. The text uses 86 different characters, which are presented in 3 different fonts for headings, main text and side notes. Some pages also contain handwritten notes, which were treated as artifacts and filtered when possible.

[3] http://kallimachos.de/kallimachos/index.php

Figure 4.1: Three sample pages from the "Narrenschiff". Pages with headlines usually also contain an illustration and follow the shown layout. Each of the presented pages showcases various image degradations, all three types of font and handwritten notes.

Since this thesis only focuses on the OCR engine, pre-processing steps were kept consistent. All 102 pages have first been binarized using percentile filter (Afzal et al., 2013), followed by the removal of all non-text elements. As a last step the text lines have been extracted from each page following a semi-manual approach. After applying the automated segmentation step from OCROpus (Breuel, 2008), the segmentation results have been manually checked. However, even after using the data set multiple times, some segmentation errors could still be found.
The use of LSTM-RNN requires a normalized height for all input images. In all application scenarios the line normalization provided in OCROpus [4], which is an implementation according to Yousefi, Soheili, Breuel, and Stricker, 2015. The line level transcription for this data set has been provided by language experts, but contained various inconsistencies especially with respect to ligature annotations and decomposed Unicode. The transcription data therefore has been transformed into decomposed normalized Unicode.

After pre-processing the data set consists of 3418 text lines. The continued correction of text line segmentation errors and using different parts of the data led to a changing number of text lines for the various experiments. The

[4] https://github.com/tmbarchive/ocropy

exact number of used text lines and pages is reported with each experiment. This is a rather small data set from a machine learning perspective but not an unusual size for a homogeneous historical data set. For most experiments the data has therefore been split in favor of a larger training and smaller validation and test sets. This is generally not a problem in unsupervised training cases, since the goal is to maximize the recognition accuracy on the data itself rather than finding the best generalizing model.

CHAPTER 5

ANYOCR: A SEQUENCE LEARNING BASED OCR SYSTEM FOR UNLABELED HISTORICAL DOCUMENTS

This chapter introduces "anyOCR", an OCR system for unlabeled historical documents. As discussed in chapter 4, OCR for historical documents comes with additional challenges. Besides the age of the document, which can lead to document degradation and damages, as well as the irregularities and artifacts caused by historic printing processes, there is also a lack of large coherent transcribed data sets needed for supervised state-of-the-art sequence learning OCR approaches. At the same time old languages, scripts and fonts often require specific expert knowledge to read and transcribe these documents, which means generating the required transcriptions can become prohibitively expensive and time consuming. All of these reasons speak in favor of working with unlabeled data, which is more readily available due to digitization efforts in the last decades.
It has already been mentioned in Chapter 2, that unlabeled or unpaired training data does not necessarily require unsupervised training approaches. Transfer learning, generating synthetic training data or semi-supervised training have been introduced as alternatives. Similarly the focus of this chapter is not to explore unsupervised training for OCR systems. Instead it tries to solve the problem by answering the question whether LSTM-based OCR can be trained with erroneous transcriptions and how they can be collected or generated.

The proposed "anyOCR" system combines state-of-the-art LSTM-RNN OCR with clustering and minimal manual annotation as an efficient way to generate synthetic transcription data. Using clustering as a semi-supervised OCR system is not completely new and has been explored previously. Junaidi, Fink, et al., 2011 combined character clustering based on shape features with manual annotation of the cluster centers. Vajda et al., 2015 further investigated many combinations of clustering techniques and handcrafted features to optimize the process. These works identified clustering as a low effort approach to apply OCR in a semi-supervised way and combined it with ensemble voting and pattern matching to minimize the error.
“anyOCR" further expands on this idea through the use of sequence learning during the final recognition by combining it with an LSTM-RNN. It also uses a different clustering with an automatic feedback and re-clustering loop, for any unsatisfactory results. The details of the clustering algorithm used in “anyOCR" are introduced in section 5.2.2

Since the proposed method is most useful for difficult to transcribe historical documents, besides a high performance it also aims for a high level of usability and accessibility. The name "anyOCR" not only emphasizes the generality of the concept, which is unspecific towards any particular language or scripts, but also for its secondary goal to be accessible. Rather than being restricted to language experts, paleographers and OCR engineers, it aims to be also usable by laymen with less or no language or domain specific knowledge. To further facilitate this agenda, a concept for layman transcriptions is introduced and it is shown how it can be used in combination with the anyOCR system. Finally, the whole system has been integrated into a web service with the same name, which is shortly introduced at the end of the chapter.

This chapter first explores the use of synthetic transcriptions to train LSTM-RNN in Section 5.1 before introducing the full "anyOCR" pipeline, from text line images to the recognized text output, in Section 5.2. Finally it will introduce some modifications to CTC based LSTM-RNN training that further facilitates the use of layman annotation in Section 5.3 and give a short overview of the “anyOCR" web service in Section 5.4.
The work presented in this chapter has previously been published as Jenckel, Bukhari, et al., 2016a, Jenckel, Bukhari, et al., 2017 and Jenckel, Parkala, et al., 2018 by the same author. This chapter therefore might contain the same wording, figures and unmarked quotes.

5.1 Training LSTM-RNN with Imperfect Transcription

Before describing the "anyOCR" pipeline there is a need to discuss the use of synthetic transcriptions compared to using synthetic images in the training data. The latter is well established within the OCR community to either generate new or augment existing training data. Jaderberg et al., 2014 proposed a automatic high quality word image synthesizer, which was used to train a competitive word recognizer for text in natural scenes. Similarly Gupta et al., 2016 proposed a method to generate training data for text recognition in natural scenes using a geometric approach, that allowed them to add realistic text sequences into generic images. Again, this allowed for the training of a high precision text recognition model. Recent work by Pondenkandath et al., 2019 also indicates, that GAN based data synthesizer could soon solve the problem of synthetic image data generation even in difficult domains such as historical documents.
While the advantages and disadvantages of using synthetic image data are quite clear, synthetic data in the target domain is a rather novel concept. The main reason might be, that if one manages to create a reasonably good system to generate text corresponding to an input image, how is that different from an OCR system? Indeed the question is justified and the answer depends on what *reasonably good* means in the context of the application. A system that generates synthetic transcriptions with an accuracy of 99% might as well be called an OCR system and no further processing is necessary. On the contrary, an accuracy of 80% is too low for many OCR demands, but still might be useful as synthetic training data for deep learning based OCR. The process of generating and using synthetic transcriptions could therefore also be seen as an iterative OCR process where initial results are reused and further refined.

In chapter 4 deep learning based OCR has been introduced as a learned mapping from the image domain to the text domain. This process happens by sampling the input space in the form of training samples and updating the OCR parameters such that prediction and desired output match. With the use of synthetic training samples either of the domains are estimated. While the image domain spans a high dimensional feature space from which the deep learning approach learns to extract relevant features, the text domain is usually low-dimensional, consisting of only the character class probability. Hence, generating synthetic image data leads to a training sample which will always correctly map to the right text domain equivalent, however due

to the high dimensional feature space, it is easy for such a training sample to become irrelevant. This happens if the synthetic images do not correctly estimate the high dimensional features of real data and introduces a mapping from an irrelevant part of the input space. Since the deep learning approach learns its own features from the input data, there also is no guarantee that the learned features are actually useful to make distinctions in the real data. A training sample with synthetic data in the target domain is guaranteed to represent a relevant sample in the input domain, since it uses real input data. It is also easier to ensure the relevancy of the sample in the target domain, due to its typically low dimension. However, bad synthetic data can lead to wrong mappings from the input to the target domain. This problem is mitigated by the fact that deep learning requires large amounts of data to correctly sample the input space, which also means that learned mappings are averaged over many similar samples. If the synthetic text data is reasonably accurate, then a lot of errors will be averaged out this way and only edge cases with few training samples are effected.
Despite the theoretical difficulties, training state-of-the-art OCR engines with synthetic image data has been shown to be successful in various scenarios, while synthetic ground truth or reusing OCR results has not been explored much. Another work in this direction is from my colleagues A. Ul-Hasan and S. S. Bukhari who in parallel explored the usage of transfer learning with Tesseract [1] to create synthetic transcriptions (Ul-Hasan, Bukhari, et al., 2016).

5.1.1 Error Types

Before testing an LSTM-based OCR's ability to deal with erroneous training data, there is some theoretical limitation to what errors LSTM training can mitigate and which it can not.
Real OCR data is not homogeneously distributed and automated approaches as well as human annotators can miss classes with few occurrences. Instead, they will merge these classes with other classes that look similar, effectively removing them from the training data. One example in the Latin data introduced in section 4.4.3, would be the character "ę" which can easily get grouped with "e" or "g". Besides the LSTM's problem to learn from few samples itself, it can also not improve on errors where a whole class was integrated into another class as it is not provided with any samples that would point

[1] https://github.com/tesseract-ocr/

towards the now missing character class.
A second very similar case are characters that are not particularly rare, but which are just generally hard to differentiate. This problem is often enhanced in historic documents, either due to low quality prints, degradation or damages. In the introduced Latin data set for example there is an old form of the character "s" written as "ſ" which can easily be confused with "f" due to their similar shapes. If these two get mixed into the same class, then the same problem arises as in the first case. This case is rather uncommon and often a problem of purely shaped based recognition or layman annotations, with a lack of language specific knowledge.
Instead, it is more common that characters are confused as similar looking characters in only some cases, e.g. due to the mentioned difficulties with historical documents. For the character classes "ſ" and "f", some "ſ" might be transcribed as "f". The same could also happen in the other direction simultaneously where some "f" might be transcribed to be an "ſ". These errors are typical for clustering based approaches where confusions happen near the cluster borders.
A third error type are completely random confusions. These confusions are typical for strongly degraded characters or scanning artifacts, where a character becomes unidentifiable and therefore could be anything. To a lower degree this also includes errors caused by clustering approaches with characters that are close to many cluster borders.
A last error type are missing or extra characters. Unlike confusions, these errors are not a mismatch between the character shape and the corresponding ground truth, but a misalignment between the input and target sequence. While this type of error is rare in human annotation, segmentation based automated approaches often face under- and over-segmentation errors, leading to extra or missing characters.
In the last three cases the LSTM is expected to learn the proper statistical features of the language and mitigate the confusions to some degree. That is under the conditions that the classes in question have a relevant number of correct samples to learn from and the percentage of confusions is small in comparison.

To compare the potential use of synthetic transcriptions, the following experiment explores an LSTM-based OCR's ability to deal with training data, whose ground truth data contains errors similar to those expected in synthetic transcriptions.

5.1.2 Experimental Setup

This experiment aims to evaluate the performance of LSTM-based OCR's performance with transcription data that contains various types of errors as described in the previous section. The OCR performance is evaluated by training multiple OCR models with various error rates in the ground truth. The introduced error is given in terms of a confusion rates $c_{ij} \in \{0, 1\}$ between the character classes i and j, describing the percentage of characters i replaced with the character j. Depending on how many classes are affected this results in a total error rate err_{total}. The types of errors that are introduced into the ground truth are:

- **1-sided confusion**: For this case a character class pair with similar shapes is selected and the characters of one class are randomly replaced by the other class's.

$$err_{total} = \frac{c_{ij}|I|}{\sum_n |N|} \tag{5.1}$$

- **2-sided confusion**: This scenario is similar to the 1-sided confusion, but the replacement happens in both directions.

$$err_{total} = \frac{c_{ij}|I| + c_{ji}|J|}{\sum_n |N|} \tag{5.2}$$

- **random-uniform confusion**: Here characters are randomly replaced by characters of another class and each character has the same chance of being replaced with any other. This approach effectively changes the character class distribution to be more uniform.

$$c_{ij} = const.; err_{total} = \frac{\sum_i \sum_j c_{ij}|I|}{\sum_n |N|} \tag{5.3}$$

- **random-proportional confusion**: This is the same as random-uniform confusion, but the chance for a character of one class to be replaced with a character of another class is proportional to that character class's relative frequency. Unlike the random-uniform case, this approach approximately conserves the character class distribution.

$$c_{ij} = \frac{1}{|J|}; \sum_j c_{ij} = const.; err_{total} = \frac{\sum_i \sum_j c_{ij}|I|}{\sum_n |N|} \tag{5.4}$$

- **realistic confusion**: Finally, the replacements are taken from ground truth generated by applying the pipeline described in the following (see section 5.2) and up- or down-scaling the relative errors.

For this test the simplest kind of LSTM-based OCR is used, consisting of a single bidirectional LSTM followed by a fully connected layer with softmax activations. The input layer of the LSTM is set to 48 and the hidden layer to 100. All text line images were resized so the height matches the input dimension. The output layer size is equal to the number of distinct character classes in the data set $checknumber$. Each OCR model has been trained with a learning rate of 10^{-4} and momentum of 0.9 using statistical gradient descent. During the training process, which lasted for 100000 steps or ≈ 38 epochs, every 1000 steps a new model has been recorded. The number of epochs is only a rough estimation, since samples were drawn fully random, so there is no guarantee each sample of the data set has been used for training an equal number of times. All reported errors have been calculated using the Levenshtein-Distance.

5.1.3 Results

For the evaluation only the last 10 models for each training process are considered. The OCR's ability to learn from erroneous ground truth is then evaluated by comparing the test error to the introduced error in the ground truth as well as the expected error. The expected error is the baseline performance of 2.5% plus the additional introduced ground truth error, if not addressed and directly transferred to the output. The baseline CER has been extracted from training with the error-free ground truth and comparing it to models trained with small total ground truth error. This resulted in a CER of $2.55\% \pm 0.25$.

For the one- and two-sided confusions specific character class pairs had to be chosen for the experiment. The chosen pairs are c-e, r-t and ſ-f. These three pairs have been chosen, since these pairs are among the top confusions in the baseline model and they all contain classes of similar shape. Additionally, these three pairs are three classes of confusion-pairs when looking at the relative class distributions. “c" is a much smaller class compared to “e" with magnitudes of 2173 and 5622 respectively. The character classes “r" and “t" on the other hand are of about the same size with magnitudes 3879 and 4333 respectively. Lastly the character class “f", with 531 instances, is much smaller than “ſ" with 1523 instances.

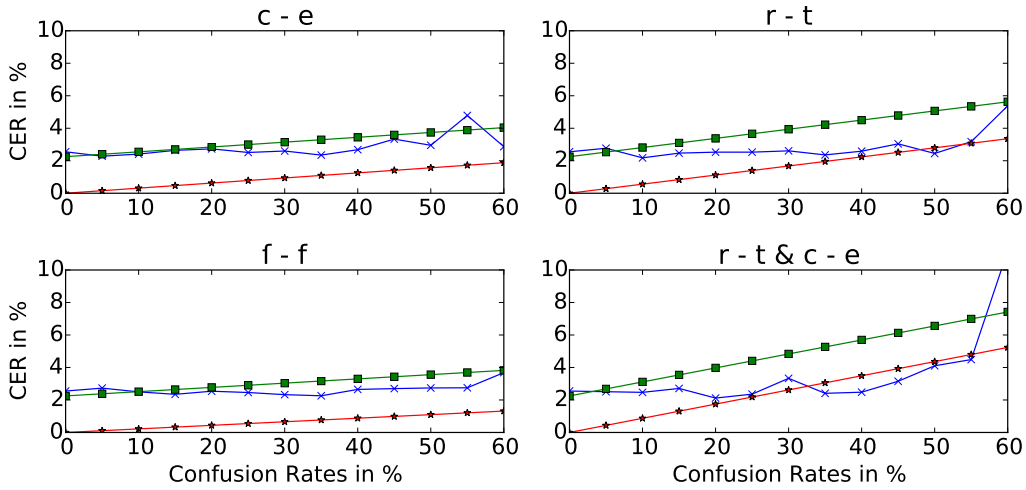

Figure 5.1: One-Sided Confusion results CER for 4 different one-sided confusions introduced to the training data. The first character is randomly replaced with the second, according to the given confusion rates. The absolute error in the training data (red), expected CER (green) and best model CER (blue) after training is compared. For almost all values the CER is smaller than the expected CER which means the LSTM could learn form the erroneous training data.

One-sided Confusions

In the one-sided confusion case, for all three pairs the resulting CER stays between the introduced and expected CER. This is partially due to the relatively small introduced errors (red line) in comparison to the baseline CER. Even up to confusion rates of $c_{stable} = 0.4$ the resulting CER stays within the expected variance of the base line performance, which makes it hard to distinguish effects of the introduced ground truth errors and normal model variance. However, this also means, that within that range, no additional errors could be observed, even though there was an additional error introduced into the ground truth. To better see the effect an additional set of models was trained using both the c-e and r-t confusion pairs. Even with this combined larger introduced error there is only a smaller jump in output error at a confusion rate of $c = 0.3$. Otherwise the output error consistently stays below the expected error and even undercuts the introduced error for confusion rates $0.35 < c < 0.55$. Overall the resulting CER seems to sharply increase at confusion rates of $c > 0.55$, which is not surprising considering

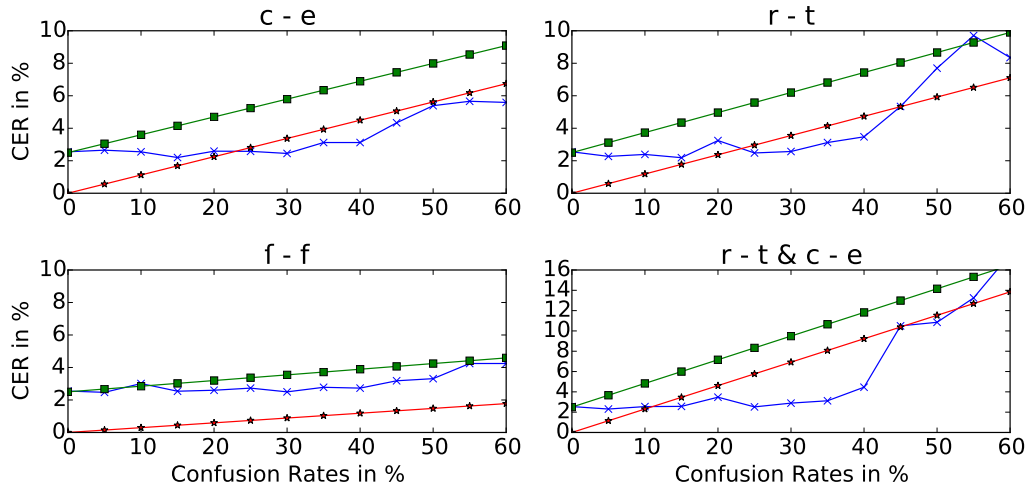

Figure 5.2: Two-Sided confusion results CER for 4 different two-sided confusions introduced to the training data. The absolute error in the training data (red), expected CER (green) and best model CER (blue) after training is compared. For almost all values the CER is smaller than the expected CER which means the LSTM could learn form the erroneous training data.

that there are more erroneous samples than real samples in the ground truth. As for the comparison between the various pairs, there either is no observable difference with respect to the different class size relations or the effect is very small.

Two-sided Confusions

In comparison to the one-sided confusion scenario, training OCR models with ground truth that contains two-sided confusions between the introduced character classes seems to behave very similar. Again, the resulting CER stays within the baseline variance for confusion rates up to $c_{stable} = 0.3$. Comparing the relative introduced error rates (red line) for the two-sided c-e experiment and the one-sided c-e plus r-t experiment indicates that this slightly lower value is not just because of an increase in overall introduced error and more likely an effect of the increased complexity of the problem. Similarly for most experiments the resulting error rate starts to increase sharply earlier as well, starting at confusion rates $c > 0.4$. What is different however, is that in most cases the resulting error not only stays below the expected error, but also undercuts the introduced error. Again the comparison between the various

pairs shows no observable difference accountable to the different class size relations.

Random and Realistic Confusions

The many simultaneous confusions introduced by both the random and realistic confusions can lead to a scenario, where the CER of the ground truth is lower than the confusion rate used to generate the data, even though the same confusion rate is used for all the character classes. This is because multiple confusions within a single text line can lead to only a single shift-error when calculating the CER with the Levenshtein-Distance. Therefore, rather than reporting the confusion rate, instead the ground truth CER is reported directly. Also, since the introduced error rates are large compared to the baseline, the expected CER is omitted and only the introduced CER is reported.
All three experiments show a consistent result. The resulting CER after training with erroneous ground truth is lower than the introduced CER. This is true, even for introduced CER of up to 0.5, where half of all characters are randomly replaced with another character. In the two scenarios with random confusions the resulting CER stays almost constant up to an introduced CER of 0.2 and starts increasing clearly only at an introduced CER of 0.4 and more. Comparing the two random confusion scenarios, the proportional confusion seems to outperform the uniform confusion especially with increasing introduced CER.
This can be understood better when looking at the character distribution of the erroneous ground truth used to train these models. When the introduced CER reaches 0.5, the character distribution after applying the two different random confusions varies greatly. While the proportional confusion closely follows the original distribution, the uniform confusion also makes the character distribution more uniform, therefore the LSTM model also learns a different underlying character distribution, leading to a worse result. This uniforming effect on the character distribution is much weaker for smaller introduced CER and therefore similar performances can be observed for small introduced CERs.
The character distribution after applying the realistic confusion also roughly follows the original distribution, but the LSTM-based OCR, trained with these kind of errors in the ground truth, performs worse. There are two main differences between the realistic confusion and the two random confusion experiments. First, there are several cases, where a class with a medium to small number of characters now has no characters. As was mentioned earlier, this type of error can not be improved on by the OCR since the corresponding

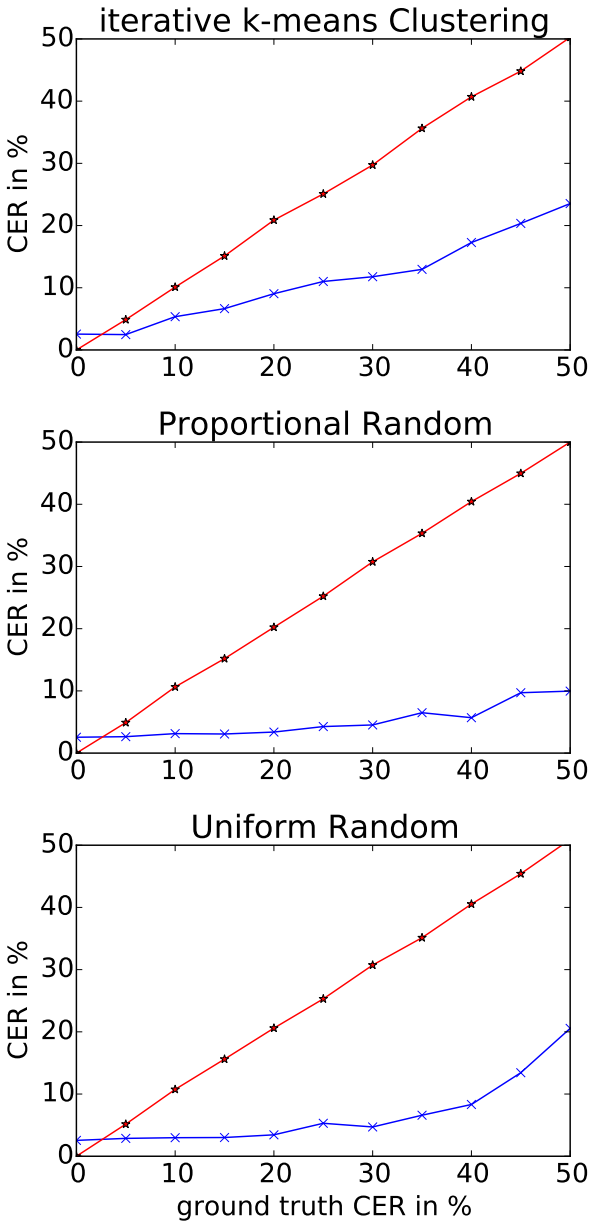

Figure 5.3: Random and realistic confusion results CER for realistic clustering and two types of random n-to-n confusions used to create training data. The absolute error in the training data (red) and best model CER (blue) after training is compared. For almost all values the CER is smaller than the expected CER which means the LSTM could learn form the erroneous training data.

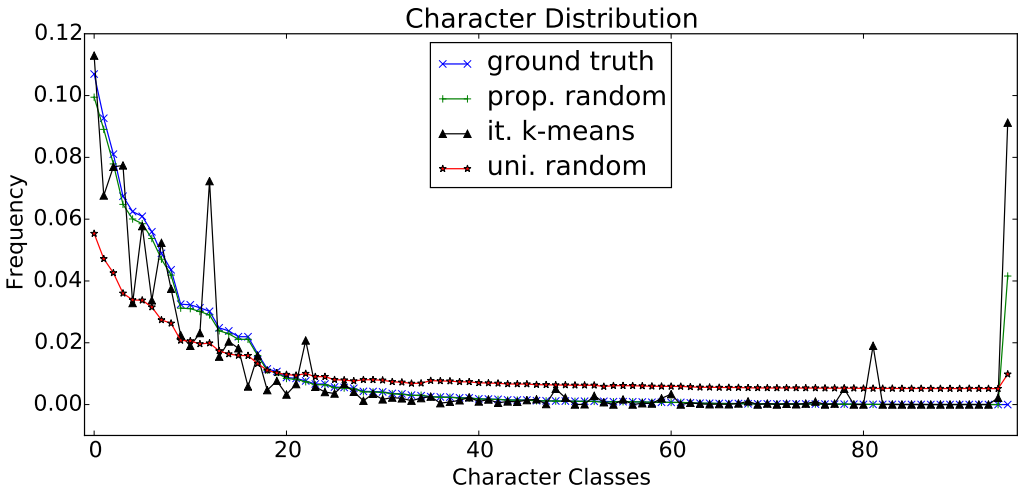

Figure 5.4: Character distributions after generating the ground truth for the uniform random n-to-n confusion at 50% CER (green), the proportional random n-to-n confusion at 50% CER (red) and the up-scaled realistic confusions from clustering at 50% CER (black) compared to the correct original character distribution (blue). The last character class represents missing characters. The closer the distribution follows the original distribution, the better are the results for training with erroneous training data.

character class is completely missing from the ground truth. Second, there is a notable spike in the realistic confusion's character distribution for missing characters. This type of error has also been modeled in the random confusion scenarios, but only in the realistic confusion scenario it is the second biggest character class with almost 10%. Similarly, although they are recorded with their respective classes, an almost equal amount of extra characters is to be expected, resulting in a very high number of alignment errors in the ground truth.

The results allow for the conclusion of three major points. First, from the comparison of one-sided and two-sided confusions it can be concluded, that training LSTM-RNN with CTC is reboust against both types of confusion up to a confusion rate of 0.4. For clustering and layman annotation based synthetic transcription generation approaches this means, that confusing characters with similar shapes, due to document degradation, grouping inaccuracies or a lack of domain specific knowledge, can be compensated as long as the confusions stay small enough. Comparing the two types of random confusions, especially at high confusion rates, the performance difference

suggests, that LSTM-RNN based OCR relies heavily on learning the correct character distribution. Additional reliance on higher order n-gram statistics can be assumed, but was hard to evaluate due to the increasing sparseness of the respective distributions. Finally, LSTM-based OCR seems to have a harder time with alignment errors in the ground truth compared to pure confusions. This is indicated by the results of the realistic confusion experiment, which showed a worse performance than in the uniform random confusion one, although the resulting character distribution followed the original one better. Overall this experiment showed, that CTC-based LSTM-RNN training for OCR can compensate for erroneous ground truth data and explored various error types, which are likely to appear in synthetic or layman transcriptions, as well as their effect on the overall OCR performance. Additionally, the importance of the validation and model-selection process on the final OCR performance has been observed. Due to the statistical nature of ANN, the model accuracy between the recorded models of a single training process can vary significantly. Training with erroneous ground truth data seems to enhance this effect and not all models will necessarily perform equal to the best quality model reported here. It is therefore still necessary to have a well transcribed validation set and a robust model selection process.

5.2 anyOCR Pipeline

The experiments detailed in the last section established the viability of using erroneous transcription for training LSTM-based OCR engines. In this section a pipeline is proposed that uses this concept by using a low effort method to generate synthetic transcription and using it to train a LSTM-RNN based OCR engine with it.
This anyOCR pipeline can be described as a combination of segmentation-based and segmentation-free OCR approaches. The segmentation-based approach is semi-supervised and is used to generate the synthetic transcription through unsupervised machine learning and minimal manual annotation. It uses segmentation to separate the text line image into single character images and clusters them into groups of similarly shaped characters. The segmentation is followed by manual annotation to satisfy the symbol grounding problem (see 2.3.1) and give meaning to the found clusters. The resulting erroneous ground truth is then used to train a state-of-the-art LSTM-RNN based OCR model in a supervised manner.
The combination of the two approaches makes use of the advantages of both segmentation-free and segmentation-based OCR, while minimizing

their disadvantages. Segmentation, clustering and annotations errors which accumulate in the synthetic ground truth can be compensated through LSTM-RNN's ability to learn and apply the sequential features of the underlying language. Segmentation-free OCR on the other hand need large amounts of transcribed training data to be usable, while the segmentation-free approach can be applied with minimal annotation effort.
In Chapter 4 a full OCR pipeline has been defined as consisting of the four steps, pre-processing, layout analysis, character recognition and post-correction. Within this pipeline "anyOCR" only deals with the character recognition step. It therefore assumes the input document images are already pre-processed and segmented into single text lines and no further post-processing is applied. This also excludes the use of any explicit language models or dictionaries in the decoding step.

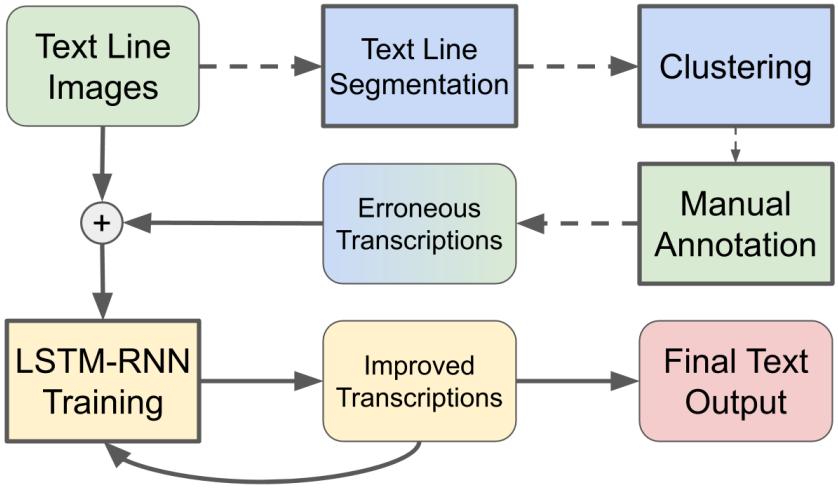

Figure 5.5: Complete anyOCR training pipeline First text lines and unique symbols are extracted from the scanned data and clustered. After the resulting clusters have been identified through manual annotation, an erroneous transcription for each text line is generated (semi-supervised, dashed). In a second phase the LSTM-RNN based OCR model is trained using the erroneous training data (supervised, full). The output of the trained model can be used to generate improved transcriptions for iterative retraining. When no better OCR model can be trained or gains become too small training is stopped.

5.2.1 Text Line Segmentation

In terms of sequence learning, text line segmentation introduces which elements in the sequences need to be labeled and allows for treating each segment individually. Segmenting text line images into single characters is part of many traditional OCR approaches, but with the rise of segmentation-free OCR is mostly avoided due to its difficulties. These difficulties include the often touching characters or disconnected character parts in many fonts, scripts or writing systems. This leads to segmentation being an additional source of errors, since segmentation errors are often directly translated into OCR errors. Historical documents can further increase the complexity of this problem through various document degradation's or low quality printing techniques.

The best performing segmentation approaches, like the one used by the popular Tesseract OCR engine [2], use a dynamic programming approach to segment text lines (Smith, 2007). The idea is to combine segmenting and labeling the characters, such that the segmentation acts as a proposal generator. Each likely dissection between characters is evaluated by the ability to identify the character within the segment between two dissections. This results in a chained dependency, where each new dissection depends on the previous one. With the help of dynamic programming, the goal is to find the set of all cuts that give the highest total likelihood of segmenting the text line into recognizable individual characters.
In unsupervised training cases this approach is not applicable, since it requires reference samples of properly segmented characters to train a classifier. Like many unsupervised machine learning algorithms, unsupervised segmentation algorithms instead make use of prior information about their application and target domain, such as how the text line was generated and what type of language and script it contains. Two basic but widely used approaches are projection profile based and connected component based segmentation. They both use the fact, that many printed books containing only Latin characters and have generally few touching characters. This means segmentation can use the white space between them to identify good dissections.
The exact choice of segmentation approach depends on the language and script in question. For the historic data introduced in Section 4.4.3, connected components have been identified to be a reliable choice for the segmentation.

[2] https://github.com/tesseract-ocr/

Connected Components

Connected components are formed by continuous connected foreground pixels in a binary image. They can be extracted by first binarizing the text line image and then assigning different labels to all foreground pixels. If pixels are connected through adjacency, their labels are unified to a single label until each connected pixel region only has a single label. Applying the connected component algorithm to a text line image does already provide a reasonable segmentation, but can be improved further for Latin scripts.
One problem are multi-component characters like i or punctuation symbols such as : or ;. Identifying the connected components will over-segmented these characters, but it can be addressed through simple heuristics, since the disconnected parts have regular shapes, e. g. dots or horizontal bars, which are above the character and have a high vertical overlap with the character's other components. Another heuristic can be used to describe the differences in spacing between characters, words or over-segmented character parts, to distinguish between the three. The first two are natural occurrences in many Latin languages, while the latter is a common problem of poor printing quality or document degradation.
Under-segmented characters, where two adjacent characters touch and are therefore labeled as a single connected component, on the other hand are more difficult to address through connected component based segmentation. However, in the context of the proposed processing pipeline they are a minor problem as can be seen later. Finally, it is recommended to filter components, which are not part of any character. Such high frequency noise, bleed through or underlines can be removed through size or shape heuristics.

After extracting the single character images, the OCR process has to give meaning to the various optical shapes that represent different characters. Without sufficient reference data, it is difficult to automatically learn this meaning through statistics or co-occurrence. Instead manual annotation is used to solve this symbol grounding problem. The required effort for this annotation can be greatly reduced if the single character images are first grouped according to their shape similarity.

5.2.2 Clustering

This type of unsupervised grouping is called clustering. The goal is to find data-inherent structures in pre-defined feature spaces to group elements that

have similarities in those features. Due to the inhomogeneous nature of the character distribution for any language, the groups to be extracted are of varying size ranging from 10% of all characters down to as few as < 5 samples. Also, since the character shapes can be highly varied and some shapes are more similar to another than others, one would also expect varying densities for these clusters, depending on the chosen feature space.
Using clustering to simplify the annotation processes for OCR is not a new approach. Junaidi, Fink, et al., 2011 used clustering as a semi-supervised approach alongside ensemble voting and manual annotations. Another approach by Retsinas et al., 2015 additionally used human supervision not only for the annotation process, but also as quality feedback to the clustering itself, discarding results which are not clearly identifiable.
The challenge was to find a robust clustering algorithm and a reliable feature space. In an earlier work by Jenckel, Bukhari, et al., 2016b various clustering algorithms have been bench-marked for Latin OCR. The surprising result is that the k-Means clustering algorithm, combined with pixel intensity features, outperformed more sophisticated state-of-the-art algorithms and character specific features. The proposed solution combines these findings into a new approach that uses the best performing approach plus an automated-feedback loop.

iterative K-Means Algorithm

The k-means algorithm is a standard clustering method introduced by Lloyd, 1982 and has been successfully applied in OCR related clustering tasks (Jenckel, Bukhari, et al., 2016b; Junaidi, Fink, et al., 2011; Vajda et al., 2015). It describes an iterative process that alternates between assigning each sample to the closest cluster center and then updates the new cluster center according to its members' average. The "closest" is defined in terms of the euclidean distance in the pre-selected feature space. Besides the feature space, the quality of the clustering result usually also depends on the initial cluster centers as well as the number of clusters k. The latter plays a much smaller role in the proposed algorithm. Since the clustering is directly followed by a manual annotation step, instead of trying to find one cluster per character class, the goal is to minimize the number of wrongly grouped characters. In its extreme this would result in one cluster per character image as it would guarantee that no character image of one class is grouped with a character image of another class.
Such a scenario would be highly unpractical and not reduce the required manual annotation effort. Instead the proposed approach balances accuracy

and simplification by starting with a slight over-estimate for the number of clusters $k > k_{real}$. The result of this initial clustering is then evaluated by an automated feedback system and iteratively applies finer grained k-Means clustering where necessary. The automated evaluation is based on how blurry the cluster center is, which is similar to how humans would evaluate the coherence of a cluster. The iterative k-Means clustering is continued until either the resulting clusters become too small or reach the desired homogeneity.

The whole process is:

- The character images are resized to a fixed size and pixel values are used as the feature space.
- Normal k-Means clustering is applied with an overestimated number of clusters $k > k_{real}$
- Check for each cluster $i; i \in 0, ..., k$, if their cluster center c_i is coherent according to a measure F
- If the coherence is below the threshold $F(c_i) < thresh.$ and cluster is not too small already $|i| > n_{min}$, a new k-Means clustering is applied to the samples in this cluster with a roughly estimated k_{new} and the process returns to the previous step

The described iterative application of the k-Means algorithm also allows to find groups of characters with very few samples that would otherwise be fully integrated into other clusters.
The coherence measure $F(c_i)$, used as the automated feedback, has the goal to measure the blurriness of a clusters center. If a cluster mostly contains members of a single class, due to the normalizing of the character position and size, the average image of that cluster will result in a slightly blurry, but clearly identifiable character image with Gaussian-like noise around the edges. Instead, if the same cluster contained a large amount of members from a different class, the average image would instead look like a mixture of both characters overlapping, similar to a bleed-through effect. This can be evaluated mathematically by analyzing the pixel intensities. Since the clustering is done on the binarized images, the intensity of a pixel in the average image directly translates to the percentage of cluster members that have that pixel as a foreground pixel. Four different types of pixel can be identified this way

- **core pixel**: Core pixel are foreground pixel that are present in at least 90% of all cluster members and make the core of that cluster.

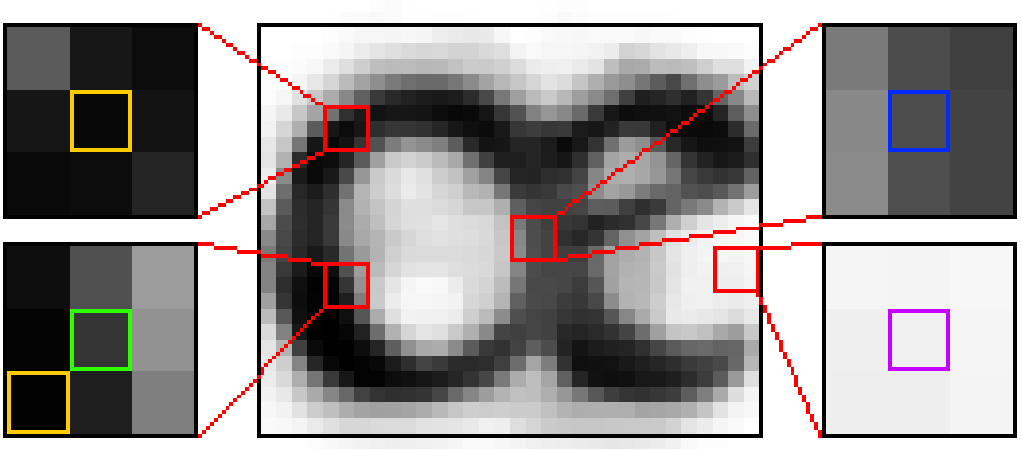

Figure 5.6: Automatic cluster evaluation Examples for every of the 4 pixel groups: core pixel (yellow), pixel in a core pixels vicinity (green), non-core pixel (blue) and background pixel (purple).

- **background pixel**: Similar to the core pixel, background pixel are those which are present in less than 5% of cluster members and therefore can be considered background.
- **noise pixel**: Even though the character images are normalized with respect of size and position, there is still a natural variety in terms of character shapes. This variety can be seen in the average image in terms of noise pixel, which are present in $5\% - 90\%$ of the members, but they are in the direct vicinity of core pixel.
- **non-core pixel**: All other pixels can be considered non-core pixel and are typically caused by cluster members belonging to a minority group or a mixture of multiple character classes in a single cluster.

The final coherence can then be defined as the rate of core pixel compared to all core and non-core pixel.

$$F = \frac{\#core}{\#core + \#non - core} \tag{5.5}$$

Similarly, the new number of k for re-clustering can be estimated using the maximums in the histogram of pixel intensities. If for example a cluster's members are 30% from one character class, 20% from another and 50% from a third, then the histogram of pixel intensities roughly shows three maximums for these values. An exact estimate however is not necessary, because in the worst case additional iterations of re-clustering can be applied.

Manual Annotation

The unsupervised generation of synthetic transcription data requires manual input at some point of the process. A system might learn to separate and recognise arbitrary symbols, but manual input is required to give meaning to these symbols. More specifically the shape of a printed letter has to be mapped to its desired digital representation, e. g. Unicode.
With all character images clustered, it is up to a human annotator to give the correct label to each of the cluster centers. Unlike normal transcription, where the text is annotated line by line, annotating only the cluster centers is much faster, but also cheaper. When looking at a full text line or words, the annotator is required to use his expertise to create high level transcriptions by interpolating uncertainty when identifying characters or words through his knowledge about likely candidates. By discarding the sequential structure of language through segmentation and clustering this information is no longer available and annotation has to be done purely based on recognizing character shapes. This also means that a language expert has a much smaller advantage in the annotation process compared to an untrained layman and might not be necessary at all.
This annotation process is also the reason why under-segmentation or extracting much more clusters than character classes is no issue. Under-segmentation creates images of character groups rather than single character images, however if under-segmentation happens often, it is likely due to systemic problems with either the language's script or the printing process. Therefore, under-segmentation is likely to reoccur with the same character sequences ultimately leading to a bigger amount of similar character group images. During the clustering process these groups are likely to be grouped together, such that during annotation, transcribing the whole sequence solves the problem. The problem of multiple clusters representing the same character class is also solved through the annotation. The annotator merges these clusters by assigning the same Unicode to them.
For the annotation process a simple UI has been created. It provides the user with a grey-scale image of the cluster average and requires the input in form of a Unicode. A more sophisticated annotation UI has been tested as well, providing additional information as well as various quality of life improvements to the annotator, however the user response did not indicate a significant improvement (Amornkosit, 2018). Also see Section 5.4 for more details on the web service.

5.2.3 LSTM-RNN Training

With the annotation results propagated from the cluster centers to their members and the positional information of each cluster member record during the segmentation process the transcription for each text line can be generated. The LSTM training then happens in the same way as with other LSTM-based OCR approaches as described in Section 3.1. However, similar to how the synthetic transcription is used to train LSTM models, their output can be used to train multiple iterations, each trained with the output of the previous one (Ul-Hasan, Bukhari, et al., 2016). Another difference is that the training data and the application data are the same. In a normal use case scenario, when applying OCR to an historic text, the OCR engineer has to identify the language and script of the historic documents and choose a suitable trained OCR model. Alternatively, the OCR engineer needs to construct a training data set and train a new model. In the anyOCR pipeline the training data, which is used to train the OCR model, and the application data, which is supposed to be OCRed, are the same. The goal is therefore to increase the OCR accuracy on the training data itself.

5.2.4 Experimental Setup

The performance of the proposed anyOCR pipeline has been tested using the 16th century Latin version of the novel "Narrenschiff", which has been introduced in Section 4.4.3. The data set has been split into 100 pages of training data consisting of 3329 text lines and a subset of 3 pages with 111 text lines for validation. The latter is used to determine which of the model snapshots performs best. The test set consists of 2 unseen pages from the same book as well as 8 pages from a different Latin book from the same time period and is used to evaluate how well the trained model generalizes. For a better evaluation of the segmentation and clustering process, anyOCR is also compared to the similar OCRoRACT pipeline, which differs in how the synthetic transcription has been generated.

As mentioned in Section 5.2.1, the connected component based clustering has been improved through multiple simple heuristics. More specifically the chosen values for this data set are:

- If a component has less than 20 pixel in total or is wider than 200 pixel, it is considered some kind of artifact and discarded.

- If there is a complete vertical overlap of two components, they are merged into one component.
- In case of a partial overlap, if the horizontal displacement is less than 10 pixel, they are also merged.
- If the horizontal distance between two components is 11 pixel or more, it is considered a space between two words.

All of these values have been chosen empirically, however for the noise filtering and the size of spaces between words, a more automated approach could be implemented. The rule for the partial vertical overlap is also more specific to the script of the data set, with its slight horizontal offset for i-dots and similar character parts.
As a feature vector the linearized pixel matrix of the character image is used. For this all segmented character images were downscaled to 32x32 and centered, resulting in a 1024-dimensional feature space. During the initial k-means clustering, the number of clusters has been largely overestimated with $k = 200$ and a threshold of $F > 0.9$ has been chosen for the re-clustering. The minimum size of a cluster for re-clustering is $n_{min} = 5 \cdot k_{new}$.
The implemented LSTM-RNN consists of an input layer of size 48, a single hidden layer of size 100 and an output Softmax activation layer of size equal to the number of classes. All text line images have been normalized to the height equal to the size of the input layer. With a learning rate of $1e - 4$ and a momentum of 0.9, the LSTM-RNN has been trained for 100000 steps using SGD. This is equivalent to training for roughly ≈ 33 epochs.
All errors reported in the following are CER calculated using the Levenshtein-Distance.

Model	TRAIN	VAL	TEST
OCRopus	7.37	6.29	10.00
OCRoRACT	9.34	6.57	10.60
Clustering	16.48	-	-
anyOCR	7.33	6.53	10.16

Table 5.1: anyOCR performance comparison CER of anyOCR, OCRoRACT, OCRopus and the clustering result. The OCRopus model performs only marginally better than the two trained with synthetic ground truth. anyOCR also outperforms OCRoRACT on the test set, indicating better generalization. OCRoRACT results taken from (Ul-Hasan, Bukhari, et al., 2016)

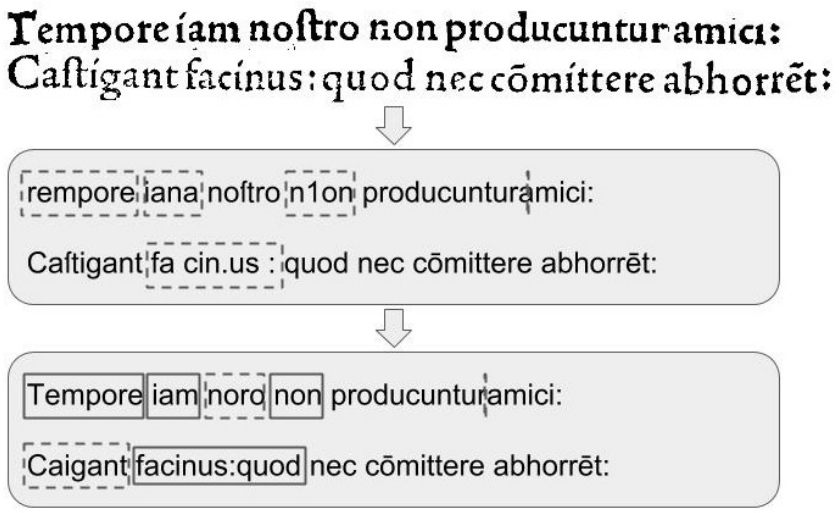

Figure 5.7: Performance of the anyOCR pipeline Two extracted sample text line images (top) and their output after clustering (middle) and LSTM training (bottom). After clustering there are a total of 8 errors. LSTM training reduces these errors to 1 and introduces 2 new errors.

5.2.5 Results

For the evaluation every 1000 training steps a new model has been recorded resulting in a total of 100 different models for each LSTM training. Only the last 30 of these models have been used in validation and only the best of these models are reported. Even though the pipeline allows for iterative re-training of the LSTM with the previous iterations output as synthetic ground truth, the anyOCR pipeline already converged after the first iteration.

To show the effectiveness of the proposed anyOCR pipeline it is compared to the results of the OCRoRACT pipeline presented in Ul-Hasan, Bukhari, et al., 2016, as well as an OCRopus model which is trained using full expert transcriptions. The OCRopus model achieves the best results out of these three approaches, however the two using synthetic ground truth have very similar performances. Especially anyOCR performs close to the baseline model, while OCRoRACT trails behind in the most important training accuracy. A good training accuracy usually indicates over-fitting, however when using synthetic ground truth, this is the data for which the model has been trained in the first place and the main goal is to improve the performance for this data. OCRoRACT trails behind significantly on this metric, although it achieves

similar generalization performance on the unseen data of the validation and test set. On a side note, the OCRoRACT went through three iterations of retraining with the previous models output while the anyOCR model did not profit from additional iterations. anyOCR even slightly outperforms the baseline model on this metric, even though due to the small sample size, the difference is not significant.
The validation and test performance of anyOCR and OCRoRACT are also very similar and only slightly behind the baseline OCRopus model. This shows that even though they were trained with synthetic ground truth, they generalize to similar data set almost as good as baseline models.

To better understand these results it is also beneficial to look at what errors exactly these three approaches produce in terms of their top confusions. OCRopus, OCRoRACT and anyOCR all produce different confusions. When looking at the results for the training data, OCRopus has one major confusion with missing characters “ſ". All other confusions are much smaller in comparison and are concentrated around other missing or added characters rather than confusions. The character “ſ" is a rather special character in this data set, as it is close in shape to the character “f" but from a language point it is closer to the character “s". It also exists as part of the ligature “ﬅ", which even in the expert annotation is not consistently annotated as such, but often instead as split characters “ſ" and “t", which can lead to many missing characters. Interestingly, the OCRoRACT results show a very different set of top confusions, which is more focused around missing and extra spaces, indicating the model could not learn the distances between words properly from the synthetic training data. On the other hand, there is no major confusion with

OCRopus			OCRoRACT			Clustering			anyOCR		
Total Errors 198			Total Errors 255			Total Errors 443			Total Errors 197		
Pred.	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**
_	ſ	50		_	15	f	ſ	38	_	ſ	38
	_	8	_		12		_	31	_		26
_	l	5	_	i	10	_		25	_	l	18
i	_	3	ā	ē	6	ﬅ		21	_	i	5

Table 5.2: Comparison of top confusions on the training data GT denotes the expert transcription, **Pred** denoted the predicted output and **Error** shows the number of errors. _ shows the *insertion* or *deletion* of the character respectively. OCRoRACT results taken from (Ul-Hasan, Bukhari, et al., 2016)

the character “ſ" and the errors are much more spread out.
At the time of this work, the OCRopus implementation showed some considerable difficulties in recognizing certain characters, especially “ſ" and “ﬅ", while later version of the engine no longer showed this problem. This also explains the considerable better results in the baseline model reported in Section 5.1. Since both the baseline and anyOCR used the same OCRopus implementation, this should not have effected the final results, but could mean a better performance compared to the OCRoRACT model.

The confusions after applying the segmentation, clustering and annotation to the data set mostly shows errors, which are typical for these processing steps. The major confusion between the characters “f" and “ſ" is either because they could not be separated properly during the clustering or the annotator could not identify the character from the cluster representative. Similarly, the missing and extra space errors are typical for a segmentation method that extracts fixed size spaces, which is often not the case for historical documents. Finally, the confusions produced by the full anyOCR pipeline are closest to the OCRopus model. While the confusions are the same, they are more spread out between the top 3 and tend more towards missing rather than extra characters. The main confusions after the clustering steps do no longer appear within the top 4 confusions, which is in line with the experiments from Section 5.1. Confusions between characters are much easier to handle for the LSTM and could be resolved to a good degree, while segmentation errors, especially those that lead to missing characters, can not be compensated for as well.

OCRopus			anyOCR			OCRopus			anyOCR		
Total Errors 181			Total Errors 187			Total Errors 702			Total Errors 713		
Pred.	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**
_	ſ	77	_	ſ	76	_	ſ	142	_	ſ	143
_	l	11	_	ﬅ	22	_		29	_	ﬅ	50
_		8	_	ę	9	_	l	20	_		38
.	_	5	.	_	16	.	_	16	_	ę	22

Table 5.3: Comparison of top confusion on the validation (right) and test data (left) GT denotes the expert transcription, **Pred** denoted the predicted output and **Error** shows the number of errors. **_** shows the *insertion* or *deletion* of the corresponding character.

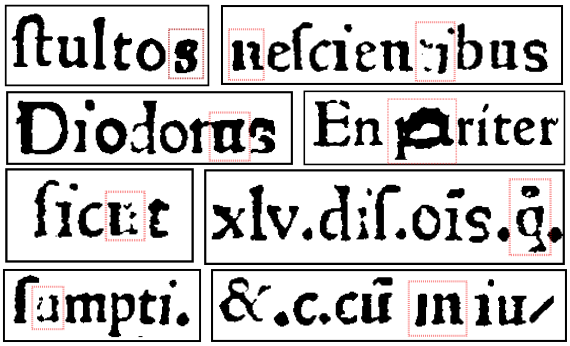

Figure 5.8: Degradation in historical documents Identifying some individual characters (red) can be challenging in the presence of document degradation or low printing quality.

5.3 Training LSTM-RNN with Fuzzy Ground Truth

One of the goals of developing anyOCR was to find a way to become less reliant on expert annotations, which are especially required for historic documents. anyOCR mitigated this advantage by relying on annotating averages rather than single characters, mostly ignore the sequential information in the language and instead require basic human skills like pattern matching. Other annotation software such as Transcribus (Kahle et al., 2017) [3] focus on giving the user all the necessary tools to efficiently annotate large data sets, but it offers no solutions on how to deal with ambiguity during the annotation process. Ambiguities are not uncommon in historical documents and can be caused by printing errors, ink spill or other document damages and degradations. In these scenarios the annotator has to rely on his knowledge about the language and script to make a decision, for example by considering the context given through the surrounding characters and words.
This section presents an alternative way to enhance layman annotations, by removing the need to make decisions when encountering annotation ambiguities and therefore lowering the impact wrong annotations have on LSTM model training.

[3] https://transkribus.eu/Transkribus/

5.3.1 Fuzzy Ground Truth

The idea behind fuzzy ground truth, is to take the ambiguity which arises within the annotation and translate it to the annotation rather than making a decision. This allows an LSTM-based OCR engine to make the decision based on its learned sequential structure rather than only the shape of the character in question. This is also because, as long as the predicted character is in fuzzy ground truth, the network parameters will not be shifted towards predicting the wrong character. Such an annotation option is especially helpful for layman, who can not rely on their language expertise during annotation and produce a higher rate of wrong annotations instead.
In Section 5.1 it was shown that training LSTM-based OCR models can mitigate erroneous annotations to some degree. So even if the annotator would create erroneous annotations, not every wrong annotation would necessarily lead the OCR engine to produce an erroneous output. Still, during backpropagation an erroneous annotation will lead to a parameter update towards the wrong output, even though the model might have predicted the correct character. These wrong updates might be rare enough to not shift the parameters of the LSTM far enough to result in the wrong prediction. It can lower the classification confidence though and lead to a wrong OCR output in edge cases.
More technically, instead of a target sequence $s = s_0, s_1, ..., s_N$ for normal ground truth, fuzzy ground truth may contain multiple possible labels at each position in the target sequence

$$s_i = (s_{i0}, s_{i1}, ..., s_{iJ}) \tag{5.6}$$

Here s_i describes the set of all possible annotations for the i-th element in the ground truth, where all s_{ij} have the same likelihood of being the correct annotation.

5.3.2 Backpropagation

In order to train an LSTM with fuzzy ground truth, only minor changes have to be made to the training process which was outlined in Section 3.1 and 3.1.2. As a reminder, LSTMs can be trained for sequence labeling tasks such as OCR by using CTC and backpropagation through time. CTC is an error function that directly evaluates how well a predicted sequences aligns with the ground truth sequence. For this CTC calculates the likelihood over all

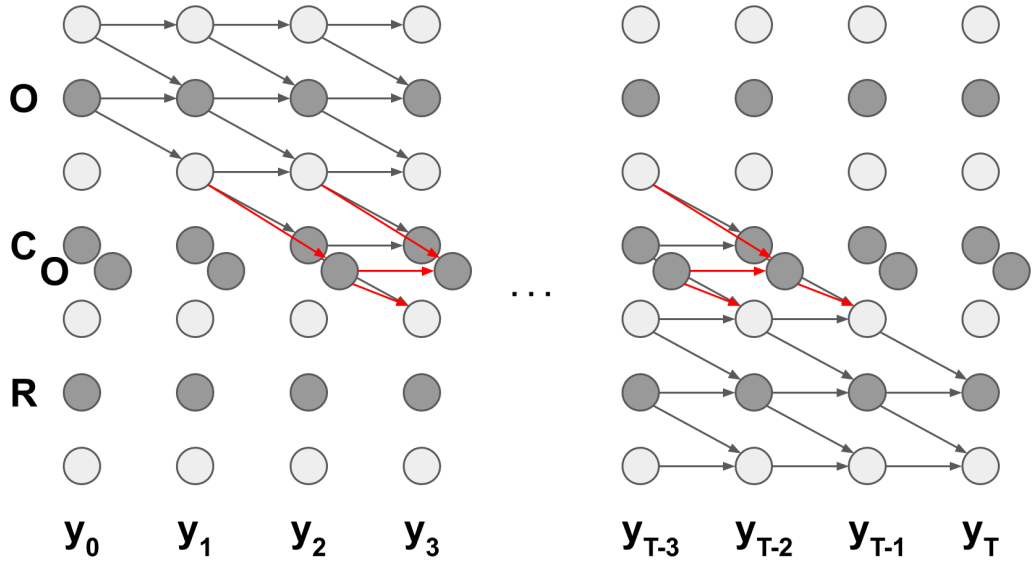

Figure 5.9: Visualization of the modified CTC-loss calculation In combination with LSTM-RNN the CTC-loss function considers all possible alignments between the network outputs (bottom) and the target sequence (left). For the forward-backward algorithm the modifications lead to additional transitions (red) to and from the multiple annotation options, while the forced blank label (grey sphere) between every two symbols in the target sequence reduces the number of possible alignments (compare with Figure 3.2).

possible, according to pre-defined rules, alignments between the character probabilities predicted at each time step and the target sequence using a dynamic programming approach called "Forward-Backward-Algorithm".

For CTC to be able to handle fuzzy ground truth it is necessary to see how the additional annotation options change the algorithm. The forward variables $\alpha^t(n)$ are defined in Equation 3.27 and consist of two terms. The first term $y^t_{s'_n}$ is the likelihood that the t-th symbol in the predicted sequence represents the n-th symbol in the modified target sequence s and directly comes from the character probability output of the LSTM. The second term describes the summed likelihood of possible predicted sub-sequences that allow for the t-th output in the predicted sequence to represent the n-th symbol in the target sequence according to M from Equation 3.24. If there are now multiple n-th target symbols n_i, then each of those could be used to calculate

$\alpha^t(n_i)$ independently, since they do not influence each other. For the $t+1$-th predicted symbol, there are now additional terms, as transitions are allowed from either $\alpha^t(n_i)$.

$$\alpha^{t+1}(n) = y^t_{s'_n} \begin{cases} \sum_j \sum_{i=n_j-1}^{s} \alpha^t(i) & \text{if } s'_n == \text{"}blank\text{"} \text{ or } s'_n == s'_{n_j-2} \\ \sum_j \sum_{i=n_j-2}^{s} \alpha^t(i) & \text{else} \end{cases} \tag{5.7}$$

but the $\alpha^t(n_i)$ are identical up to the factor $y^t_{s'_{n_i}}$, as they had the same preceding sub-sequences. With that it is possible to re-define

$$\alpha^{t+1}(n) = \sum_j y^t_{s'_{n_j}} \sum_{i=n-1}^{s} \alpha^t(i) \tag{5.8}$$

such that $\alpha^{t+1}(n)$ describes the total likelihood over all annotations given for the n-th symbol in the target sequence. The secondary condition, which allowed for different characters to directly follow each other without a "blank" character between them has been removed here, since the condition $s'_n == s'_{n-2}$ is ambiguous if either of the two could have multiple annotations. This does not restrict the functionality of CTC and is used per default by some CTC implementations such as the one in OCRopus [4].
For the backward variable it is analogous to re-define

$$\beta^t(n) = \sum_j y^t_{s'_{n_j}} \sum_{i=n+1}^{n} \beta^{t-1}(i) \tag{5.9}$$

With these changes CTC-based LSTM training can be applied as described in Section 3.1.2 when using fuzzy ground truth.

5.3.3 Experimental Setup

Due to the high manual effort of annotating a full training data set of $100+$ pages it is difficult to test the performance of fuzzy ground truth in a natural application scenario, comparing expert and layman annotations. Instead two separate experiments have been conducted. One shows that training with fuzzy ground truth works in principle and the other introduces a more practical application using fuzzy ground truth with the anyOCR pipeline.

[4] https://github.com/tmbarchive/ocropy

Scenario I

In the first experiment a layman annotation is simulated using synthetic data. This means, characters that are commonly found within the top confusion training on the expert ground truth as well as characters that are commonly annotated wrongly using the anyOCR pipeline, are chosen randomly and paired with the respective characters they are mistaken for. A full list of character confusion pairs can be seen in Table 5.4. The assumption is that an expert would not make these mistakes and identifies the correct annotation from the pair every time. A layman on the other hand would have difficulties with making an informed decision and therefore by guessing only identifies the correct annotation for 50% of the pairs. Besides the two training data sets created this way, a third data set is generated with fuzzy ground truth, containing all the chosen annotation pairs. This setup limits the number of fuzzy ground truth options to two per target character.

Scenario II

In the second experiment fuzzy ground truth has been combined with the previously introduced anyOCR pipeline. After the segmentation and clustering of the data set following the setup described in Section 5.2, the annotation task was given to 16 participants, who each annotated a fourth of the cluster centers. This resulted in a total of 4 different annotations, which were then combined into fuzzy ground truths. To be comparable to the first scenario only two annotations were combined at a time, leading to a total of 6 different fuzzy ground truths.

In both scenarios the same model and training setup has been used. A single layer LSTM with an input size of 48 and a single hidden layer of size 100

p P	l t	l 1	l r	l i	l ſ	l $/$	t r	a ā
n m	n u	1 i	c e	a o	a u	b h	f ſ	

Table 5.4: Common character confusions when transcribing Latin script. All confusions have been repeatedly observed in both manual transcriptions and character clustering. Every entry describes a bi-directional confusion between two character classes.

followed by an output layer with softmax activation. Each model was then trained for 100000 steps which equals ≈ 33 epochs with SGD, a momentum of 0.9 and a learning rate of $1e-4$. Out of the different models recorded every 1000 steps, only the last 10 were considered for validation. The experiments were conducted on the historical data set introduced in Section 4.4.3, with a similar split into 100 pages with 3263 text lines for training, 3 pages with 111 text lines for validation and 2 pages with 103 text lines for testing.
To compare the various trained models their performance is reported in terms of CERs that calculated with the Levenshtein-Distance, when compared to the available export annotation.

5.3.4 Results

Scenario I

In the first scenario three different models were trained using three different training data sets. The expert data set is the expert-annotated training data set and is considered to have no errors in its ground truth. The non-expert data set has a synthetically modified ground truth with half of all possible errors. The confusion rate has been fixed to 5% for all pairs and has been applied symmetrically to both characters. Finally, the fuzzy ground-truth contains all possible errors, but always contains both annotation options.
This results in respective CER errors in the ground truth of 0% for the expert data, 3.3% for the non-expert data and 6.7% for the fuzzy ground truth data. For the latter it is important to note, that CER counts an error whenever it differs from the expert annotation, hence all pairs lead to a full error. As expected the error rate of the fuzzy ground truth is therefore about twice has high as for the non-expert annotation.
For the validation data set, which is a subset of the training data set, as well as

	$T0$	$T1$	$T2$
$Expert$	0	2.9	2.9
$Non-Expert$	3.3	4.2	3.8
$Fuzzy$	6.7	2.7	3.0

Table 5.5: Scenario I results Comparison of the Character Error Rates (CER) for the LSTM-RNN training with perfect ground truth from an expert, guessing (50%) from a non-expert and fuzzy ground truth containing two transcription options.

the test data set, the results for both the expert and non-expert cases are what is expected. With the expert data set state-of-the-art accuracy on this data set is achieved, while the non-expert leads to a slightly worse performance of about 1 CER higher. This is in line with earlier results from Section 5.1 and once again highlights the LSTMs ability to handle erroneous transcriptions during training.
The fuzzy ground truth data on the other hand, which contained both options, also achieves close to state-of-the-art performance. The LSTM could learn the correct decision between the two options and translate this knowledge to unseen data.

Scenario II

Although fuzzy ground truth performed well in the first scenario, the conditions were still artificial with only small and homogeneous character confusion pairs in the fuzzy ground truth. The second scenario tests fuzzy ground truth in a practical application. In this scenario testing on a separate test data set has been skipped, since the goal is not to create the best generalizing OCR model, but to generate the best OCR for the training data.
The four layman annotations have a relatively high CER in the range of $19.7 - 27.5$. After training and validation the CER on the training is reduced by about 4 - 5 points, which is in line with the previous finding. After pairing each annotation with each other to generate a total of six fuzzy ground truth transcriptions the best CER is improved to $14.3 - 17.7$. The best CER for fuzzy ground truth here describes the CER that compares with both annotation options and counts the one with less errors. This way a wrong annotation can

Annotation	1st	2nd	3rd	4th
1st	19.7	-	-	-
2nd	$17.1(5.1\%)$	27.5	-	-
3rd	$15.4(19.4\%)$	$14.3(17.1\%)$	20.5	-
4th	$17.7(8.1\%)$	$17.7(5.1\%)$	$14.7(15.3\%)$	23.4

Table 5.6: Combining multiple annotations to create fuzzy ground truth
The diagonal elements show the CER for each layman annotation, while the off-diagonal elements are the best CER when combining two annotations, creating fuzzy ground truth. The disagreement (brackets) is the percentage of fuzzy character annotations in each training data.

Annotation	1st	2nd	3rd	4th
1st	14.7	-	-	-
2nd	14.6	23.9	-	-
3rd	14.9	14.3	15.3	-
4th	14.1	14.1	14.5	19.7

Table 5.7: Results of combining multiple annotations to create fuzzy ground truth The reported CER results after LSTM OCR model training. The diagonal elements show the CER when training with each layman annotation, while the off-diagonal elements show the results after training with the created fuzzy ground truth transcriptions by combining two annotations.

be paired with a correct annotation to improve the best CER of the combined fuzzy ground truth.

The disagreement between the different pairs is another indicator that describes the percentage of characters with multiple ground truth options. This does not mean, however, that one of these options is the correct annotation an expert would give. Especially the combinations with the third cluster annotation have a high disagreement while also providing the lowest best CER. Even though the third annotation itself is not particularly accurate, it contains many different annotation errors than the other three annotations.
After training and validation, the training CER of all fuzzy ground truths is remarkably close in the range $14.1 - 14.9$. The least improvement compared to training with the single annotations are observed with the first annotation, which also had the lowest CER before training. Only one of the six models trained with fuzzy ground truth performed worse than the non-fuzzy option. In the other five cases using fuzzy ground truth improved the overall performance on the training data set.
Interestingly, the combination of the two worst annotations in terms of CER and the lowest disagreement resulted in a better CER after training with fuzzy ground truth, while the combination of the two best annotations with the highest disagreement led to the worst performance. This could be an indication that fuzzy ground truth containing the correct annotation is beneficial to the training, but fuzzy annotations containing only wrong options lower the overall performance.

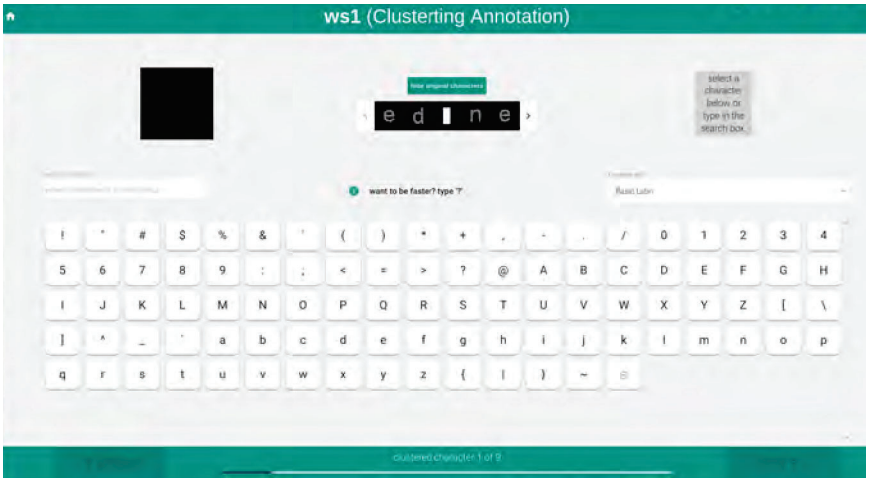

Figure 5.10: Cluster annotation UI in anyOCRWeb The user can give the Unicode for the displayed cluster representative (top left) or use the virtual keyboard (bottom). As additional information he can browse cluster members and their local context (top middle) and see the previous (top right). Taken from Amornkosit, 2018

5.4 Web Service

As a final step to improve the user accessibility of the anyOCR training pipeline it has been integrated under the name “anyTrain" into the web service “anyOCRWeb" (Bukhari et al., 2017). It has been realized mainly by Khan, 2018, who worked on the code integration, and Amornkosit, 2018, who designed and evaluated an improved GUI. “anyOCRWeb" is focused mainly on applying pre-trained OCR models and various pre- and post-processing tools alongside a backend user- and task-management system. With the addition of the described anyOCR training pipeline as “anyTrain" it also offers options for semi-supervised OCR model training.

To use the web service a user first has to create a user account and log in. After selecting an existing or creating a new work space, which store the user’s various projects, he can choose to create a new OCR pipeline or train a new OCR model. The training options allow for fully supervised and semi-supervised training with optional page segmentation, depending on whether the training data is provided as pages or text lines. After uploading the data and starting the pipeline, the backend processor will independently go through all processing steps until it has to wait for manual inputs.
In the semi-supervised scenario with text line data, the server performs text

line segmentation and clustering before prompting the user to start annotating the cluster centers. The user can either enter the Unicode codes or select the character from a virtual keyboard with previously uploaded character samples. Additionally, the user is provided with sample context-windows showing the surrounding characters of random cluster members. This is meant to help with identifying clusters that contain mostly over-segmented character parts and therefore should not be annotated.
After annotation the results are given back to the server and parameters have been set, the final OCR training is started. If a validation data set is provided the system also performs model selection. As of the writing of this thesis the anyOCR web service is not publicly available.

5.5 Summary

The main goal of this chapter was to answer the questions on whether LSTM-RNN based OCR can be trained with erroneous transcription data and how this data can be collected or generated efficiently. For this purpose first the training of LSTM-RNN with erroneous ground truth data has been explored. A selection of errors, which are likely to appear in synthetic transcriptions, have been introduced into an expert transcription and their effect with respect to the overall OCR performance has been investigated. The results have shown that standard LSTM-RNN based OCR shows good capabilities to handle erroneous transcription data during the training process. Almost all trained models have performed above the expectation and successfully corrected some of the introduced errors
With the usefulness of synthetic transcription data established, a semi-supervised low effort and accessible OCR training pipeline has been introduced. This pipeline uses unsupervised clustering and minimal manual annotation as an efficient synthetic transcription generator and combines it with an LSTM-RNN based OCR model. The pipeline allows for the training of competitive OCR models as was shown on historical documents from the $15th$ century.
In order to further increase the accessibility of LSTM-RNN based OCR model training, a new concept for fuzzy transcription has been introduced. According to this concept the gap between expert and layman transcriptions is reduced by allowing the annotator to transfer uncertainties during the annotation process to the transcription. It has been detailed, how the resulting fuzzy ground truth can be used to train standard LSTM-RNN based OCR models. The use of fuzzy ground truth has proven to be better than guessing

and has in some cases achieved results close to the baseline of expert level annotations.
Finally, it has been shown how to combine the anyOCR pipeline with this new concept through the merging of multiple layman cluster annotations. The results show the possibility of using multiple erroneous transcriptions to further improve the overall OCR performance.

Although the main objective of this chapter was reached by showing the effectiveness of using erroneous transcription data and the performance of the combined transcription and recognition pipeline, the two minor goals outlined at the beginning of the chapter could not be reached. They are both related to name of the proposed "anyOCR" pipeline and its usability. Even though it is conceptually widely applicable, it practically needs some improvements for applications on documents with more complex languages or writing systems like Chinese, Japanese or Arabic. Arabic script is highly connected and requires a different segmentation approach. Chinese and Japanese on the other hand are both languages with thousands of different characters and clustering based grouping would lead to thousands of cluster, many of which would only have few members. Manual annotation would no longer be efficient for these languages. It might be necessary for such applications to cluster on the sub-character level or conceive a completely different approach on how to generate the synthetic training data.
In the same sense, while the proposed methods successfully lowered the need for language specific expertise, they did not remove it. The first experiments in Section 5.2 with a knowledgeable annotator, who was familiar with the Latin alphabet and also possessed some basic knowledge about the Latin language, showed results that were significantly better, as compared to the later experiments in Section 5.3 with participants who only non-natively knew about the Latin alphabet and had no knowledge about the language. Similarly, this was also reflected in the average annotation time, which was about 2 to 4 times longer for the second group. This could also have been enhanced by the second group's unfamiliarity with the annotation UI, but is unlikely to be the only reason.

With the topic of this thesis being the use of OCR in unsupervised training cases, it is important to note, that the proposed methods of this chapter are not yet applicable in such a scenario. The anyOCR pipeline does generate its own synthetic training data, but the evaluation and more importantly the model selection process, is still reliant on available transcriptions. The model selection has been noted to be especially important for the training with

erroneous ground truth. It has been observed, that the quality of the trained OCR models can vary considerably during the training process and not all trained models are an improvement over the initially erroneous transcription data. For this reason, in the next chapter the focus will be on how model selection can be performed without access to transcribed data.

CHAPTER 6

TRANSCRIPTION-FREE OCR MODEL EVALUATION

The last chapter introduced the use of erroneous transcription data for LSTM-RNN OCR model training in unsupervised training cases and showed the effectiveness of training LSTM-RNN with such data. While no manual transcriptions have been used during the training process itself, a small amount of expert transcriptions were still needed as validation data. In order to decide when training is finished and which model snapshot is the best, the models performance had to be tested on expert annotations. This step is especially necessary to prevent the trained LSTM network from modelling the errors in the synthetic ground truth rather than modelling the real language properties. The main question the work in this chapter tries to answer is therefore ”How can trained OCR models be evaluated without available transcription data?". The evaluation of trained OCR models is a problem not just for unsupervised training cases, but also for most OCR application scenarios. During the training of an OCR model with a transcribed data set, a part of the data can be set aside and serve as a reference for the trained model's performance on unseen data. This is an important indicator for unwanted over-fitting and also serves as a reference, which model to choose for specific application scenarios. It is however, also the only reference, because there is no other method to directly compare the performance of different OCR models besides manual evaluation. Although this is a common problem, there has been little to no research in this direction in the field of OCR.

In other document analysis domains like NMT, where more progress has been made towards unsupervised training approaches, transcription-free model evaluation as well has become more important. Lample et al., 2017 needed

a transcription-free evaluation method to perform model selection for their unsupervised NMT approach. Their proposed evaluation uses the standard BLEU score (Papineni et al., 2002) combined with cycle consistency. In the context of translation usually both transformation directions are relevant, which allows directly applying so called “back translation" by completing a full translation cycle. They showed that unsupervised model validation done with their approach was comparable to human evaluations.
The equivalent of NMT in the image domain is a rather young field called *image-to-image translation*. Here similar approaches have been introduced that use cycle consistency to evaluate and train models in an unsupervised manner. Translating images between domains is especially difficult, because input and target domain are often only loosely specified concepts and hard to define in the context of pixel values. None the less humans can usually distinguish a black dog from a black cat in an image, even though the difference might only be in a few pixels.
R. Zhang et al., 2018 use the fact that ANN classifiers have achieved near human level classification accuracy and evaluate the quality of generated images through the comparison with existing domain samples. An exact target to compare with usually does not exist, but when processing both images with a trained ANN classifier they show similarities in the low level features. The resulting Learned Perceptual Image Patch Similarity (LPIPS) distance works almost independently of how the ANN was trained and provides similar to human evaluation.

The transcription-free model evaluation method for OCR introduced in this chapter follows these general approaches. The main difference compared to the established approaches in NMT is caused by the different modalities between the input and output domain in OCR. Since OCR transforms images into text, a different metric is required when using the concept of cycle consistency. Additionally, translation tasks naturally require a model that can transform between both languages in either direction. OCR models are one-directional and an additional model is required that performs the inverse operation of rendering text line images from OCRed text input. In this sense the proposed approach is similar to the LPIPS distance, in that it uses an ANN to transform the output into a state where it can be compared with a reference. It is shown that using such a text line rendering network allows for an transcription-free evaluation of the primary OCR model.

The chapter is structured as follows. First the details of the encoder-decoder architecture are introduced in Section 6.1. Section 6.2 then describes how to

use this architecture to perform the transcription-free model evaluation and discusses the differences compared to the standard CER based evaluation. The work described in this chapter has been published by the same author and this chapter may therefore contain unmarked quotes from Jenckel, Bukhari, et al., 2018.

6.1 Architecture

The encoder-decoder architecture used in this work follows an approach similar to auto-encoders. The trained OCR models act as encoders that map image domain input data to the corresponding OCRed text. Additionally, a secondary model takes the role of the decoder and is trained to render the OCR output back into a text line image.

For the OCR a basic LSTM model is chosen, which consists of a single BILSTM layer followed by a fully connected output layer with Softmax activations. The text rendering model also consists of a single layer BILSTM, but followed by a fully connected layer without any activation function. With this architecture the rendering model outputs a single pixel column of the target image at every time step. For an easier communication between the two models, the input and output dimensions for both encoder and decoder are chosen to match. For the same reason, instead of transferring the OCRed text between the two RNN, the predicted character probability distributions from the Softmax activations are used.

While using LSTM networks to create text line images has not been a research focus, there has been some previous work. For example Graves, 2013 explored the predictive capabilities of LSTMs for generative tasks such as generating handwritten text samples. In their proposed method an LSTM is trained to predict the pen tip positions during online handwriting. The trained network can then be primed with handwriting styles and generates new real-looking handwritten text lines.

Unlike those works, the architecture described in this section is not a generative model and its goal is not to generate new text line images, but to recreate specific ones. Directly using the Softmax output as the decoder input also means the rendering model does not require predictive capabilities

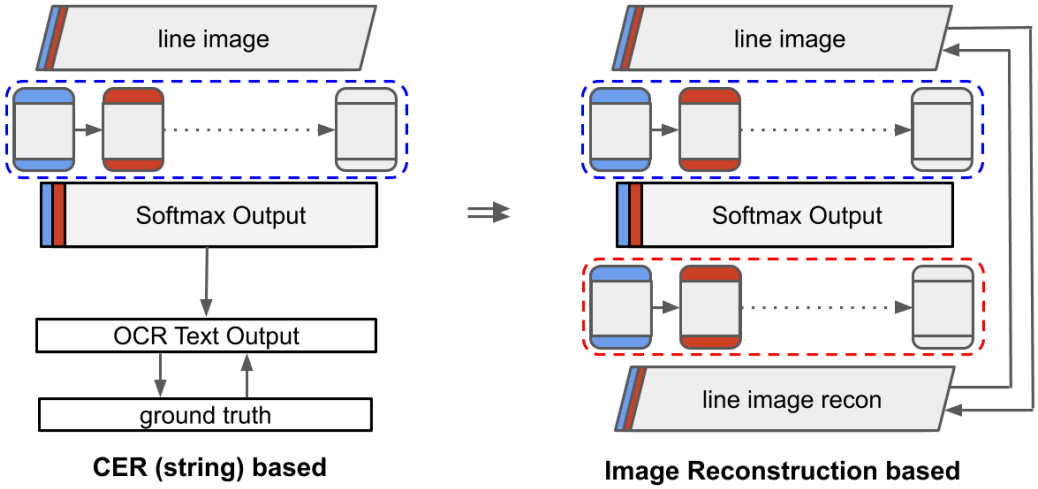

Figure 6.1: Schematic of the CER based and proposed OCR model evaluation The OCR performing BILSTM-RNN layer (blue) is extended with a secondary BILSTM-RNN (red). Instead of decoding the Softmax output into a text string for CER based comparison with the ground truth, the secondary network reconstructs the input image for an image reconstruction based evaluation.

but can learn a 1-to-1 mapping. Additionally, there is no need to learn an inverse alignment between the OCRed text and the character positions in the target image. Finally, it also mitigates the problem of learning a mapping from a discrete to a continuous space. In OCR the input image can contain many different variations of a single character, like different sizes, positions, shape variations and so on, and all of them have to be mapped to the same character class. The distribution of a single character is therefore continuous in the input domain, with endless possible variations on the pixel values, but discrete in the target domain, where a character is either a or not a. Mapping from many different pixel value combinations to a single character class is straight forward. The inverse direction would require to map from that single character class back to all the different character variations in the image domain, information which has been lost during the encoding process. The character probability distribution output by the Softmax activation however, is still continuous and theoretically able to encode specific shape variations. To prevent the OCR network to focuses on encoding shape information rather than predicting character class labels, the training procedure for the two models should encourage the encoder to focus on OCR accuracy, but allow the decoder to create distribution specific character shape variations.

6.2 Relative OCR Model Ranking

Unlike auto-encoders, the described encoder-decoder architecture is not trained jointly, but in two separate steps. The goal for the OCR model has to be unchanged in producing the highest possible OCR accuracy. It is trained normally with CTC loss and backpropagation.
For the training of the rendering model a pretrained OCR model is used to generate character probability distribution output sequences for each text line image, which are then used as the model input. Training is straight forward and minimizes the sum of absolute difference (SAD) between the predicted and the original text line image as the training objective. Practically, this training can be streamlined by combining encoder and decoder into a single architecture, but keeping the encoders parameter fixed during decoder training.
The trained rendering model is then used to evaluate different OCR models by combining them into an encoder-decoder stack and providing it with unseen text line samples. Any OCR error by the encoding model is propagated to the decoder and results in a reconstruction error of that character in the generated text line image, such that comparing it with the original image input will be a measure of the OCR model's performance.

One important note is that the evaluation provided through this process is not absolute, like the standard CER accuracy, but rather a relative performance between different OCR models. This means results from different evaluations with different rendering models are not comparable to each other. The chosen architecture consists of two separate networks and both will contribute to the total error. The rendering model will also never be perfect and achieve 100% accuracy, partially because noise and other artifacts are not recognized by the OCR model and the information about them is not forwarded to the decoder. If all OCR models are evaluated using the same rendering model however, then these errors become systematic and symmetric for all tested models. An absolute evaluation would require knowing the error of the rendering model and measuring that would again require transcribed training data, which defeats the purpose of this approach.
For the same reason the OCR model used during the training of the rendering model can not be evaluated using that same rendering model. There might be a bias in the rendering model towards the OCR model that was used during training. If no extra OCR model is available, the rendering model could also be trained using all the OCR models that are supposed to be compared. This way any bias towards a specific model should be averaged out.

For the evaluation itself again the SAD is used as a metric. It effectively measures how similar the shapes and positions of characters are between the two images.

6.2.1 Character Error Rate vs Character Shape Similarity

Comparing the proposed evaluation metric with the CER results in some theoretical differences in how OCR errors are measured and what is considered an OCR error. Even though the CER serves as a standard metric in OCR related tasks, it has some non-intuitive properties. It is arguable that the proposed image based evaluation using SAD captures the performance of the OCR model in more detail.
The most prominent difference is how both metrics assign error scores to a wrong character prediction. In terms of CER any character that does not match with the ground truth is counted as a full error, independent of what character was predicted instead. A measure based on character shape similarity instead assigns smaller errors on predictions which are wrong, but similar in shape, like an “f" being predicted as an “ſ". From a character recognition perspective assigning smaller errors to confusions between similarly shaped characters also makes sense, since in edge cases such a model would be more likely lead, to a correct prediction. For a human reader as well, a OCRed text with “e" instead of “ę" would still be quite easy to read. A model that wrongly predicted an “e" as a “W" instead, is unlikely of correctly predicting an e in any edge case.
The same is true for the inverse direction, where a character is labeled correctly, but the recreated character shape somehow differs from the original. The CER again does not take this into consideration and assigns no error, since the predicted character label is correct. Since the SAD based evaluation uses the character probability distributions as the input, a disturbed shape likely indicates that the prediction of the OCR model has low confidence and is more likely to become a wrong prediction in similar scenarios.
Another rather small disadvantage of using CER as an evaluation metric originates from its reliance on manual annotations. Even though expert transcriptions are usually treated as the 100% baseline, they can still contain some amount of mistakes or inconsistencies. The latter is especially true if multiple experts were involved. Such inconsistencies include different Unicode choices for certain characters or different annotation decisions with respect to ligatures. This can also negatively impact the quality of a CER based performance evaluation.
On the other hand there are also downsides to using character shapes as the

OCR-pred	GT	IMG-pred	Input	CER	SAD
)	M	)	M	1	0.95
ō	o	ō	o	1	0.653
ſ	f	f	f	1	0.566
f	f	f	f	0	0.453

Table 6.1: Comparison of CER and SAD based model evaluation metric for different error types CER assigns the same error whenever the OCR prediction (OCR-pred) does not match with the given ground truth (GT), the SAD is more fine grained and assigns smaller errors to similar predictions (row 2, 3) and higher errors for correct, but low confidence predictions (row 4).

basis for evaluation, which is due to their size inequality. Larger or capital characters will record larger errors simply because their increased size also leaves more room for variations in their shape. A normalization would require knowledge about the exact position of each character though and is impractical. A shape based metric therefore should be interpreted as a measure of how much of a text line is correctly recognized with respect to the size of the input image and not with respect to the length of the target sequence, which is used for CER based evaluation.

6.2.2 Experimental Setup

Transcription-free model evaluation is useful in any unsupervised training scenario, but one of its main application domains is historical documents with its smaller data sets and increased complexity to create high level transcriptions. In this experiment standard LSTM OCR models have been trained with the historical data described in Section 4.4.3, which has been split into 2673 text lines for the training set, 396 text lines for the validation set and 349 text lines for the test set. It is then followed by a model evaluation using both the proposed as well as the standard CER approach. Transcription inconsistencies have been mentioned as a possible error source and the transcriptions used for training have been provided by experts. Due to a lack of expertise, the quality of the transcriptions could not be quantified and is therefore assumed to be correct.
Model evaluation is usually used to track training progress online and determine when to stop training. Comparing model snapshots in an offline manner instead, leads to the same overall result in selecting the best model

snapshot, although it is a bit less practical. With the proposed method a reference trained OCR model is required during the training of the decoder and an offline comparison becomes more practical, while producing the same results as the online one. For this reason a single LSTM OCR model with the described single layer BILSTM architecture has been trained on a total of 100000 text lines or about 33 epochs with a learning rate of $1e-4$ and a momentum of 0.9. After every 1000 text line samples a model snapshot has been taken and the last 11 snapshots have been used for the evaluation. More precisely, the 11th last to the 2nd last models were evaluated, while the last model snapshot was used for the decoder training. During the second training phase, the parameters of the last recorded OCR model were used and kept fixed as well as the training rate and the momentum, but since the rendering model itself can not be tested, gradient normalization from Pascanu et al., 2013 as been added. This prevents instabilities in the training process, but also slows it down considerably, such that the secondary rendering model was trained for 250000 iterations to guarantee the model has converged fully.
Both the OCR model and the rendering model use the described single layer BILSTM architecture. The hidden layer is of size 100 in both networks, while the input layer of the OCR network and the output layer of the rendering model share a size of 48, that corresponds to the height all image text lines have been normalized to. The fully connected output layer of the OCR network and the input layer of the decoder share the same size of 86 as well. It is defined by the number of different symbols in the data sets' alphabet.

6.2.3 Results

To better capture the differences between the proposed approach and the standard method, the results are analyzed both quantitatively and qualitatively. The qualitative analysis verifies the different properties of each method as described in Section 6.2.1. Afterwards the quantitative analysis shows how these differences translate to the model selection process and how each approach ranks the trained OCR models. Even though they have qualitative differences, quantitatively they produce similar results.

Qualitative Analysis

The qualitative results confirm the expected behaviour of an image similarity based metric as compared to CER. Most importantly, wrong OCR label predictions are shown to directly translate to distortions in the text line image

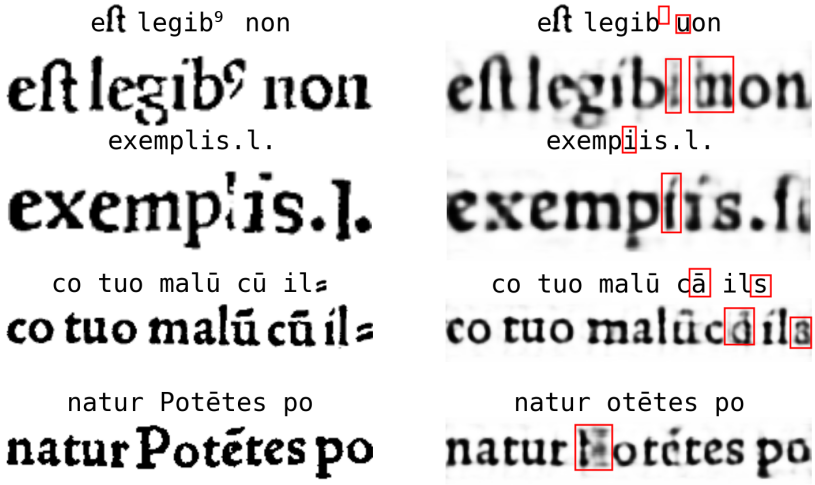

Figure 6.2: Linking OCR errors to image reconstruction errors For four text line images and their corresponding ground truth (left), a generated text line image and the OCR output used to generate it are shown (right). All OCR errors correspond to distorted characters in the generated text line images (red boxes). The particular shape of each distorted character depends on the decoder's learned context and the encoder's Softmax output.

rendered by the decoder. The recurrent network in the decoder architecture however prevents that in the rendered text line the wrongly predicted character can be seen instead of the correct character. Instead, the learned sequential structure competes with the erroneous error caused by the OCR miss-classification and seems to lead to an average of both characters instead. The high SAD error in the character region indicates that the error is correctly identified.

Since the text rendering network uses the character class distributions as input, there also is an expected effect of low confidence classifications. The low confidence leads the decoder to also be less confident about which character to render, resulting in a more blurry and low intensity prediction of the character shape. This is also recorded as an increase in the SAD error. In the historical data used for this experiment low confidence predictions were common especially for the character classes "f" and "ſ". These two character classes also are a good example to show, that confusions between similarly

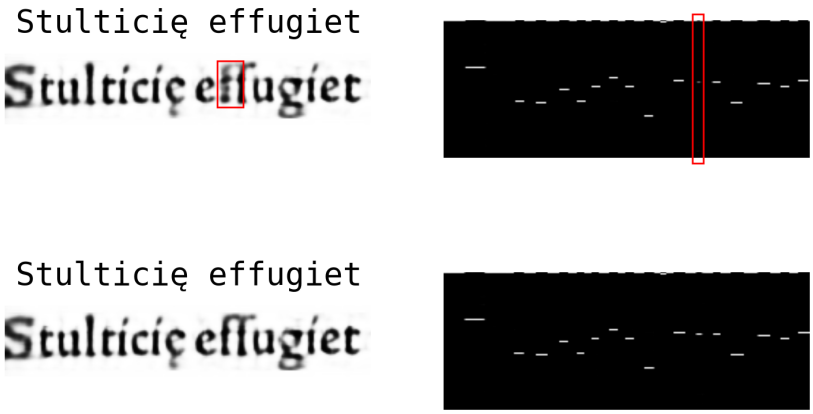

Figure 6.3: Effect of low confidence predictions on image reconstruction Comparison of two text images generated with two different OCR models. Both models generate the same OCR output, but in the generated text images we see a weak f (left, red box). This corresponds to a lower Softmax output for the character f, which is barely above threshold (right, red box). In the bottom case a stronger character in the generated image (left) corresponds to a higher Softmax output.

shaped characters score smaller SAD errors.

Another mentioned difference is how the encoder network is not encouraged to conserve information about noise and other unwanted image artifacts in its output. The reconstructed text line image therefore is expected to be mostly noise free. The experimental results confirm this behaviour for all tested models. Since it is symmetric for all models, the resulting SAD error caused by this difference in the images does not influence the relative ranking. The contribution of this systematic error to the total SAD error has not been measured.

The last observation is an increased SAD error towards either end of the text line image. This effect is not explained by the property differences of the used metric and is more likely a finite size effect caused by the RNN architecture. At the start and the end of a text line less sequential information is available about previous or following characters respectively. This should lead to an

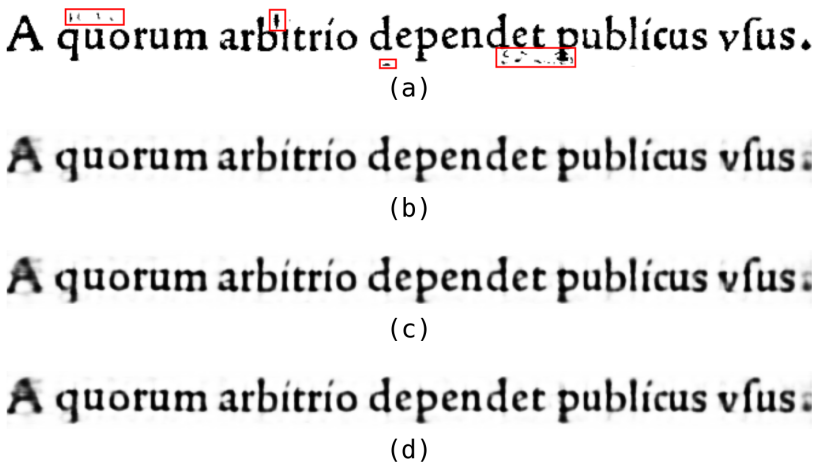

Figure 6.4: Effect of noise and other artifacts on SAD as an OCR metric The noise in the original text line image (a) marked in red, is not represented in the OCR model's Softmax output. Image (b), (c) and (d) were generated using the same decoder combined with three different LSTM-RNN OCR models. In all three cases the noise is missing.

effect similar to a low confidence prediction, but is independent of the OCR model and therefore also does not influence the relative ranking.

Quantitative Analysis

The different qualitative properties of both methods also lead to different evaluations of each OCR error. It is therefore not expected that their ranking of the evaluated OCR models is similar or that they agree on a best model. Each method follows a different logic that comes with different advantages and disadvantages. With that in mind, the quantitative results are unexpectedly similar.
With the goal of performing model selection, the evaluation was focused on the differences in the performance on the validation, rather than the test data. On that data set the absolute error rates calculated through CER are smaller than the corresponding SAD errors, but for both measures the values stay within roughly the same interval of 0.55 and 0.65 respectively. Within these intervals the values are also similarly distributed as is shown by the almost equal variance for both methods, with 0.03 for the evaluation with CER and

0.05 for the SAD one. The resulting ranking of the evaluated OCR models also does not show too many differences.
If the models are ranked according to their SAD errors and compared to ranking with the CER, then models 2, 5 and 8 are ranked differently. The other models follow the same order. While SAD ranks model 5 higher, model 2 and 8 are ranked lower. However, both rankings agree on the best model, although the CER ranking identifies two models with equally good performance.
As a last step of model evaluation, the best model during validation is chosen as the final model and once again evaluated with unseen test data. When doing this with the two models with best performance during CER based validation, model 8 achieved a CER of 2.653% while model 9 got a CER of 2.575%. So anyone using the trained model in future application scenarios would likely chose the model with better test accuracy and therefore choose the same model with either evaluation approach.
The fact that two models achieved the exact same performance in terms of CER is relatively unlikely and might be an indication that a larger validation set is needed. On the other hand it could also be interpreted such that SAD is a more sensitive metric than CER, with its non-binary evaluation of errors.

Model #	norm. SAD	CER
#0	9.161	3.101
#1	8.987	2.760
#2	9.210	2.870
#3	9.055	2.921
#4	8.781	2.740
#5	8.796	2.941
#6	8.776	2.720
#7	8.629	2.569
#8	8.666	2.559
#9	8.556	2.559

Table 6.2: Error rate comparison on the validation set. The SAD error has been normalized with the total number of pixels in the image. For both evaluations the error rates for all 10 OCR models stay within about the same interval and have a similar variance.

Rank	1	2	3	4	5	6	7	8	9	10
SAD	#9	#7	#8	#6	#4	#5	#1	#3	#0	#2
CER	#8,9	-	#7	#6	#4	#1	#2	#3	#5	#0

Table 6.3: OCR model ranking Ranked according to SAD errors and CER on the validation set. Both methods show the same best model and mostly share the same order, with the exception of only 3 models.

6.3 Summary

In this chapter a transcription-free model evaluation for LST-RNN based OCR model training has been introduced. Similar to previous works in related domains, it makes use of the cycle consistency principle by extending the basic OCR architecture with a secondary text rendering network. In the context of changing the evaluation domain from text to image, various advantages and disadvantages for either evaluation have been discussed and confirmed through experiments. The full evaluation process has been described in detail and demonstrated with a historical data set, one of the prime application domains for this kind of method. In this context it could be shown, that while the properties of the evaluation metrics differ significantly, both the proposed and standard evaluation method produced similar results with respect to the relative ranking of models. This includes, that both agreed on the same best model in a model selection process.

While the proposed evaluation method satisfies the main objective of this work, there are some areas that need further improvement. This includes the partial blurriness of the generated images, which increases the overall SAD errors and reduces the metrics sensitivity to OCR errors. Another major restriction for the wide application of this approach is the required access to the Softmax layer's outputs, since OCR models generally only provide the OCRed text. Both of these limitations have been attempted to be solved by Sinha et al., 2019 through the use of GANs. Although the approach only saw limited success due to problems in learning the alignment between the input text and the output image, other work by Pondenkandath et al., 2019 have shown promising results for historical document generation with GANs. Replacing the character probability distributions in the input layer would also mean, that the method loses some of its properties, such as its sensitivity to low confidence predictions.

With an auto-encoder like architecture and the ability to identify OCR errors through cycle consistency errors in the image domain, the next logical step would be to use transcription-free data not only during the evaluation, but also during the training phase. This could either lead to a fully unsupervised training approach or a semi-supervised approach that mitigates the need for large training sets. In the next chapter different ways of training such a method are explored and one possible direction for future research on unsupervised OCR is outlined.

CHAPTER 7

UNSUPERVISED OCR

The unsupervised training of high performance machine learning models is the goal of any machine learning field. Such approaches provide a high level of application flexibility and require low effort compared to supervised ones. In contrast the low number of established unsupervised training techniques for sequence learning is a testament for how difficult it is to design such training schemes.
In the field of OCR little to no work has been done to explore unsupervised training of RNNs. This is partially, because for the longest time OCR research has focused on areas where paired training data is easy to collect and supervised training sufficient. On the other hand the input and output domain, while sharing underlying language features, have significant differences. These include the spatially extended features of character shapes in the image input versus the purely temporal text sequence in the output domain, or the continuous pixel values and corresponding high level of variance compared to the discrete character symbols.

So far this thesis has focused on how to mitigate the lack of transcribed training data when training OCR models in order to make use of supervised or semi-supervised approaches. The recurring theme of using efficient manual input still means that these approaches could become prohibitively expensive. In this chapter the focus will be on how OCR models can be trained with mono-domain training data. For this purpose first the work presented in Chapter 6 on transcription-free model evaluation is extended to explore how the standard LSTM-RNN based OCR training with CTC loss, as described in Section 3.1, can be improved through additional unsupervised training. The prospect of improving any OCR method through the use of unlabeled training data is especially interesting in the context of historical documents, where

collecting and transcribing large data sets faces additional challenges. The results of this exploration highlights the difficulty of unsupervised OCR and the need for novel approaches to the problem.

The second part of this chapter in Section 7.2 outlines a possible direction for future developments on the topic of unsupervised training for OCR. The outlined approach is similar to recent work on unsupervised training in related domains and relies on the use of adversarial training. Similar to GANs introduced in Section 3.2 this still requires data from both input and target domain, but it is not paired into input-target samples. Instead the adversarial approach learns and aligns underlying language properties from the mono-domain data. The proposed approach serves as an outlook and comes with a set of limitations that prevent widespread application in its current form, including historical documents.
The work presented in this chapter has partially been published by the same author and might therefore contain unmarked quotes from Jenckel, Bukhari, et al., 2019.

7.1 Unsupervised Pre-Training

In the previous chapter an encoder-decoder architecture was proposed for the transcription-free evaluation of OCR models. The encoder in that architecture is simply the OCR model and with the decoder's ability to reverse the OCR operation, it allowed using cycle consistency as the evaluation criteria. While both the encoder and the decoder were trained separately in a supervised manner, the question naturally arises, whether such an architecture could be used to train the OCR model in an unsupervised manner as well. The combined encoder-decoder architecture combined with a cycle consistency criteria, at least in theory, would allow for such a training approach. Similar to auto-encoders, the question remains whether the self-learned encoding space would prove useful for an OCR task.
Even though early experiments with such an architecture show, that pure unsupervised training this way does not incentivize the network to learn OCR like encoding, it still opens up the opportunity to explore the use of transcription-free training data as a mono-domain source of additional information.

One way of using this information is *unsupervised pre-training*. Pre-training as a concept was already mentioned briefly in Section 2.3 in the context of transfer learning. It is a popular way of transferring knowledge to a network by initializing its parameters with those of a trained model. Unsupervised pre-training follows the same idea, but rather than using pre-trained parameters from other models, it learns them through unsupervised training on additional unlabeled training data. This has the goal of learning inherent features of the data, that will also be useful for the following main training objective. The goal can also be phrased as finding a good parameter initialization, which starts the training process close to a local minimum.
The most common form of unsupervised pre-training is based on cycle consistency and uses an architecture similar to the one described in the last chapter. It is also called *input reconstruction* and was introduced by Bengio et al., 2006. In particular, they have shown, that a meaningful initialization for deep CNNs can be achieved through layer-wise unsupervised pre-training. Other similar pre-training objectives for CNN try to enforce the learned features to be invariant towards certain variations in the input. Larsson et al., 2016 introduced a color invariance by modifying the objective to reconstruct gray scale versions of the input image. Similarly Noroozi and Favaro, 2016 introduced spatial invariance by randomly shuffling image patches in the input, that had to be reconstructed into the original image.
For RNN, similar work has been done by Ramachandran et al., 2016 who introduced layer-wise unsupervised pre-training for a two-layer Sequence-to-Sequence (Seq2Seq) model (Sutskever, Vinyals, et al., 2014). A major difference of RNN compared to CNN is the added temporal component, which opens the unsupervised pre-training up for additional prediction objectives. Srivastava et al., 2015 added a predictive task to the pre-training, where the second half of a video had to be predicted from the first half, and showed that it leads to learning of more robust features. More recently this has been extended to what is known as predictive coding (Lotter et al., 2016). Rather than forwarding learned features from layer to layer in multi-layer RNN, instead only the difference from each layers prediction is forwarded. Successful applications can be found for image classification, speech recognition or NLP (Oord, Li, et al., 2018).

In the field of OCR little work has been done regarding unsupervised pre-training. An exception is the work by Sahu and Jawahar, 2015, who proposed a Restricted Boltzmann Machine (RBM) as an unsupervised feature extractor. This work however only pre-trains the feature extractor, but ignores the RNN in the process.

In this section four training schemes using mono-domain training data are introduced and explored. Each training schemes aims at directly pre-training the LSTM in a single layer LSTM-based OCR model and uses some combination of pre-training, multi-task training or fine-tuning combined with reconstruction or prediction objectives. The results show the difficulty of applying any type of unsupervised training to RNN-based OCR models.

7.1.1 Architectures

The main architecture used in this section is the one introduced in the previous chapter (see Section 6.1). For comparison a second architecture based on a Seq2Seq model is evaluated as well. Even though Seq2Seq models are not usually used for OCR tasks, there has been sufficient previous work to show that unsupervised pre-training is useful for this kind of architecture. The Seq2Seq architecture will therefore serve as a baseline regarding the effect of unsupervised pre-training.

OCR with CTC-loss

The state-of-the-art LSTM-based OCR model consists of one or more BILSTM layers, often combined with a CNN feature extractor (Breuel, 2017). The model introduced in Section 6.1 is the simplest model of this type with only a single BILSTM layer and no feature extractor. To be more in line with the described standard OCR model in Breuel, 2017, the encoder has been extended by an additional CNN feature extractor and the output layer has been replaced with a 1-D convolutional layer, which improves processing speed slightly.
The first and most forward way to train the encoder-decoder architecture is to treat it as a single auto-encoder and combine it with a input reconstruction objective L_1-loss. After pre-training the encoder OCR is then fine-tuned on paired input-target samples in a supervised manner. This scheme is called “CTC-pretrain".
A secondary scheme uses the same architecture, but rather than splitting training in two distinct phases, it simultaneously trains the architecture with both CTC-loss and the L_1 reconstruction objective. Since the CTC-loss will be calculated from the encoder output, it does not influence the decoder. The

model will be trained with both transcribed and transcription-free training data in an alternating fashion. This leads to a combined objective function

$$Loss_{total} = L_{CTC} + L_{rec} \tag{7.1}$$

during the training step with transcribed data. This type of multi-task learning is reported to improve the model's ability to generalize. This scheme is called "CTC-multi".
The next scheme combines the first and the second one, by first pre-training the model using unlabeled data and then fine-tuning it with multi-task training. The third scheme is called "CTC-pretrain-multi".
The last model training follows the procedure used in the last chapter more closely. Both the encoder and the decoder are trained in a supervised way using the available transcribed training data. Here, the input during the decoder training is the output generated from the trained encoder. The goal is to use the image reconstruction errors to inform the OCR network about necessary parameter changes, since it has been shown that OCR errors are linked to image reconstruction errors. Both networks are combined into a single auto-encoder, which is then fine-tuned with unsupervised training and an L_1 reconstruction objective. The decoder will be fixed during this phase, to make sure it is the OCR network that corrects any image reconstruction error. This last training scheme is called "CTC-interval".
Besides these 4 schemes, there has also been attempts to combine the reconstruction objective with a prediction objective for either of the described models. However, early results have shown that the described architecture does not have very good predictive capabilities and those attempts have been exempted for further evaluation.

OCR with Seq2Seq

While the conservation of the sequential order between input and output domain can be used when performing OCR, in related fields like NMT this conservation is a problem. A Seq2Seq model solves this problem through the use of an encoder-decoder architecture (Sutskever, Vinyals, et al., 2014), similar to the one proposed in the previous chapter. It uses RNNs for both the encoder and the decoder, but rather than transmitting an encoding at every time step, instead the whole sequence is condensed into a single encoded feature vector. This feature vector is not one of the network outputs, but the last hidden state of the encoding network, which is used to initialize the hidden state of the decoder. This also leads to a more complex decoding

process, since the whole sequence has to be decoded from a single initial state. Rather than predicting the whole output sequence from scratch, instead the decoder treats the decoding as a series of state transitions. For each time step it takes the previously predicted output and estimates the likelihood for the next output in the sequence, based on its stored history.
For the Seq2Seq model in this work, both the encoder and the decoder are chosen to be single layer bidirectional LSTM-RNNs. In order to transform the decoder output into character probabilities, it is followed by a fully connected layer with Softmax activations.
Even though LSTM can model long range dependencies, encoding a long text line image sequentially into a single hidden state is challenging and the quality of the encoding becomes worse the longer the input sequence is. The previous architecture prevents this by forwarding the whole output sequence to the decoding network and therefore providing new information about the encoded text line at every decoding step. An alternative solution to this problem is an attention model as introduced by Bahdanau, Cho, et al., 2014. It allows the decoder to learn where to focus its attention in the encoder's output sequences at any step in the decoding process. As a side effect this also solves any alignment problem between the encoder and the decoder. The attention model has been implemented as described in Wojna et al., 2017, without adding the additional spatial encoding.

The unsupervised pre-training scheme for this architecture is in accordance with Srivastava et al., 2015. During pre-training the encoder is combined with two additional single layer BILSTM-RNN decoders who are trained to reconstruct and predict respectively. The reconstruction objective again is to minimize the L_2 loss between the first half of the text line image and the output of the reconstruction network. Simultaneously the prediction network has the objective of predicting the second half of the text line image. Here again an L_2 loss is used, resulting in a total training loss

$$Loss_{total} = L_{rec} + L_{pred} \tag{7.2}$$

Due to this multi-task pre-training, the encoder needs to encode not just the content, but also enough contextual information, so the predictor is able to make reliable predictions.
For the decoder training a Cross-Entropy-Loss between the one-hot encoded text sequence and the predicted character probability sequence has been chosen. This type of loss performs better with discrete target labels like characters, but for a better generalization these target labels have also been

smoothed (Wojna et al., 2017) with a smoothing rate of 0.1

7.1.2 Experimental Setup

Since historical documents are again one of the prime application domains, the experimental evaluation of both architectures uses the historical "Narrenschiff" data (see Section 4.4.3). For the first architecture the encoder and decoder use the same parameters as previously introduced, which means the size of the input layer is 48, the hidden layers are of size 100 and both networks were trained using SGD with a learning rate of $1e-4$ and momentum of 0.9. The size of the final encoder layer again corresponds to the size of the data set's alphabet plus one for the additional "blank" output. The training time has been increased to 250.000 steps for each training phase. This became necessary after applying gradient normalization to reduce training instability in the decoder network, which slowed down the learning process.
The Seq2Seq architecture is analogous and also uses an input size of 48 for the encoder, while all hidden layers have been chosen to be of size 128. The size of the decoder's input layer however is equal to the size of its output layer, which corresponds to the Fsize of the data's alphabet. This is necessary to apply teacher forcing (Williams and Zipser, 1989) during the training, which speeds up the decoder training by introducing the correct output of the last time step as the new input. Since the decoder learns state transitions to estimate each character in the sequence, a wrong prediction at time t would otherwise result in learning the wrong state transitions at time $t+1$. While it is common to introduce a new symbol to indicate the end of a sequence, instead the decoder always predicted 50 characters, which is the maximum length of text lines in the training data. Any additional predicted character past the corresponding ground truth was ignored during training as well as evaluation. The Seq2Seq model as well has been pre-trained for 250.000 training steps, but the supervised fine-tuning only converged slowly so it has been extended to $4.000.000$ steps. The training was performed with SGD optimizer, a learning rate of $2e-4$ and a momentum of 0.9.
Due to problems with heavy over-fitting the rate with which teacher forcing is applied has been reduced by 0.1 every 25.000 training steps. Additionally a slight weight decay of $4e-5$ has been added to the SGD training.

The historical data set has been split into 2673 lines for training, 349 lines for validation and 745 lines for testing. The training data has then further been

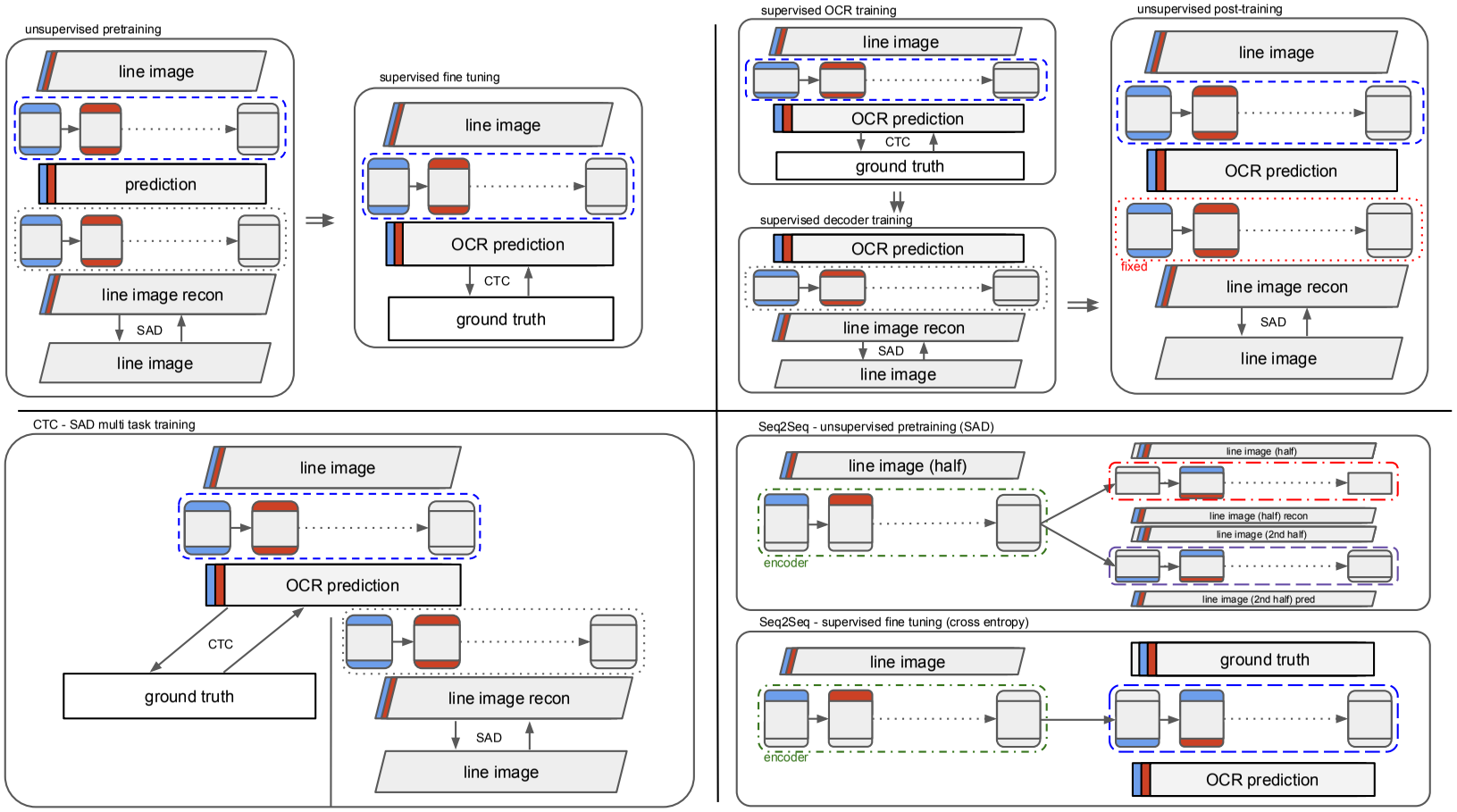

Figure 7.1: Training scheme overview “CTC-pretrain" (top left), “CTC-interval" (top right), “CTC-multi" (bottom left) and “Seq2Seq-Pretrain" (bottom right). Each block defines one training phase and all displaced networks are single layer BILSTM-RNN, with the OCR performing LSTM-RNN highlighted in blue. Not shown are “CTC-baseline", “CTC-pretrain-multi" and the baseline "Seq2Seq" training.

split into 1373 unlabeled and 1300 transcribed text lines, which are used for any unsupervised and supervised training respectively. Each text line image also has been binarized and height normalized to fit the input dimensions of the respective networks.
The relatively small size of the training data lead to an over-fitting problem with the Seq2Seq architecture. While the described attempts to mitigate the problems improved the situation training was very unstable and eventually diverged. For this reason a secondary synthetic data set has been generated, by randomly concatenating three images from the MNIST data set into one text line image (LeCun et al., 1998). This synthetic training data set is about 13 times as big with 20.000 training lines and has 1000 lines each for validation and testing.

7.1.3 Results

For the evaluation of how additional training on unlabeled training data affects the performance of RNN-based OCR approaches, the performance of each training scheme is compared with a base line model which only performs OCR and has been with the transcribed part of the training data. For the standard OCR approaches trained with CTC-loss, this is the extended encoder described in Section 7.1.1. The baseline Seq2Seq model uses the normal encoder-decoder architecture, but skips any pre-training steps.

Model	CER
CTC-baseline	3.77
CTC-pretrain	4.44
CTC-hybrid	5.46
CTC-pretrain-hybrid	13.18
CTC-interval	7.48 (div)

Table 7.1: OCR results for CTC-based training. Direct unsupervised pre-training showed the best results, but could not improve performance over the base-line. The interval training quickly diverged during the unsupervised fine-tuning.

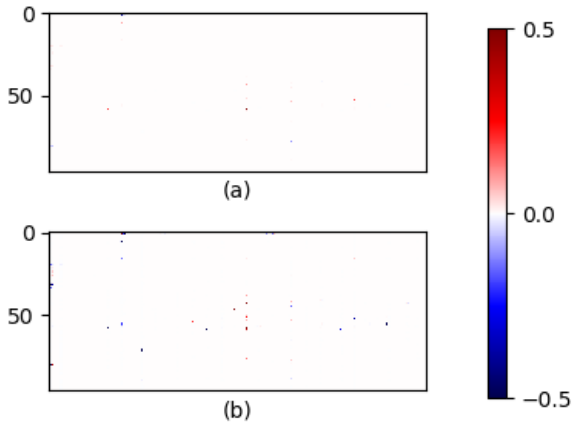

Figure 7.2: Comparison of backpropagated errors during fine-tuning. After training both encoder and decoder individually in the “CTC-interval" scheme, the backpropagated reconstruction error during fine-tuning (b) is similar to the CTC-error (a), but also contains additional errors due to noise, image artifacts and variations in the character shapes.

7.1.4 OCR with CTC

Out of the various described training schemes the simple pre-pretraining one “CTC-pretrain" performed the best with a CER of 4.44. In comparison to the baseline model without any pre-training, this performance is still a bit worse by about 0.7 points. A closer look at the learned activations for the CNN feature extractor and the LSTM-RNN layer of the encoder after and before the different training phases reveals, that the activations in the CNN layer from before and after pre-training only differ slightly from the activations after baseline training, while the differences at the LSTM layer activations are much bigger. After pre-training the activations at the LSTM layer are more homogeneous with a lower variance. The activations after baseline training

p	0.5	0.4	0.3	0.2	0.1
CTC-baseline	3.77	4.43	5.05	6.01	8.08
CTC-pretrain	4.44	8.01	7.08	8.61	11.45

Table 7.2: Effect of labeled to unlabeled training data on unsupervised pre-training For no percentage p of labeled samples compared to all samples did the unsupervised pre-training improve the baseline performance.

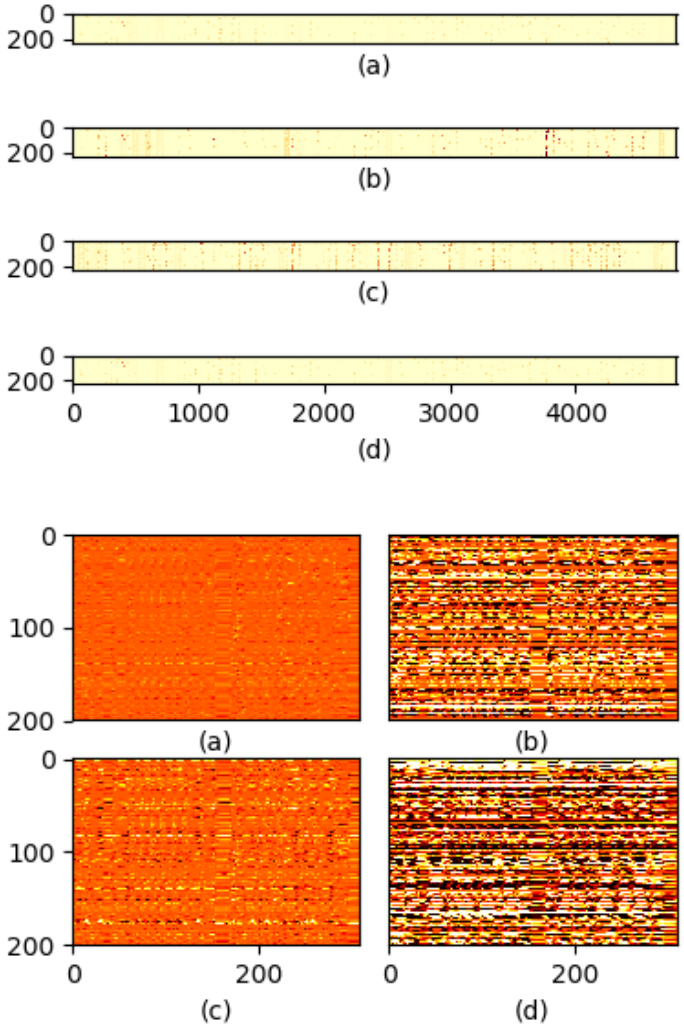

Figure 7.3: Network activations after CTC-based OCR training The shown activations are from the CNN (left) and LSTM (right) layer of the OCR encoder for “CTC-pretrain" after unsupervised pre-training (a) and after fine-tuning (b), for “CTC-multi" (c) and for “CTC-baseline" (d). Activations after pre-training are sparse (left) and homogeneous (right) compared to those after supervised training with CTC loss. “CTC-multi" (c) showns balanced activations between those of the reconstruction (a) and CTC (d) objectives. “CTC-pretrain" after fine-tuning (b) unlearned most of its pre-trained activations (a) and is more similar to those after “CTC-baseline" training (d).

as well as after fine-tuning both show a much higher variance with clearer horizontal and vertical patterns representing transitions between character shapes in the input image. Even though the fine tuned model produces activations much closer to that of the baseline model, the effect of the pre-training is still visible in the form of smoother transitions between the local extrema. The multi-task training leads to some kind of compromise in terms of LSTM layer activations between the pre-train and baseline approaches which reflects the simultaneous training with both objectives. Finally, the last approach "CTC-interval" showed an inverse behaviour compared to the pre-trained model. Shortly after the combined unsupervised training started, the encoder network lost its OCR capabilities and quickly diverged.
As a last step the influence of the ratio between labeled and unlabeled training data has been investigated, to see if a larger amount of unsupervised pre-training compared to supervised fine tuning would lead to a better performance than the baseline. For this the best performing "CTC-pretrain" model has been trained with progressively larger amounts of unlabeled and smaller amounts of labeled training data, starting at a $50/50$ split like in the previous evaluation. The results again show that pre-training can not improve model performance above the baseline, which at the end was trained with only 10% of the data.

All of the results indicate that there is no positive or synergistic effect of unsupervised pre-training for CTC-based OCR with an image reconstruction objective. Even though the performance of the pre-trained model is only slightly worse than the baseline, it is more likely that the fine-tuning process was long enough, for the model to forget its pre-training. After the fine-tuning the model was unable to reconstruct any input image and instead only produced basic sequential features. However, the initialization prevents the model from reaching a better local minimum. The simultaneous training with both objectives in the multi-task approaches prevents the network from forgetting what it has learned during the pre-training. The model is able to reconstruct the input image after training, but at the cost of a lowered OCR performance. Again the initialization with the unsupervised pre-training seems to lead to a bad local minimum. The interval training approach, where supervised and unsupervised training are reversed, shows the inverse behaviour of the unsupervised pre-training approach. This time the trained OCR encoder forgets its learned features during the final unsupervised training and diverges.

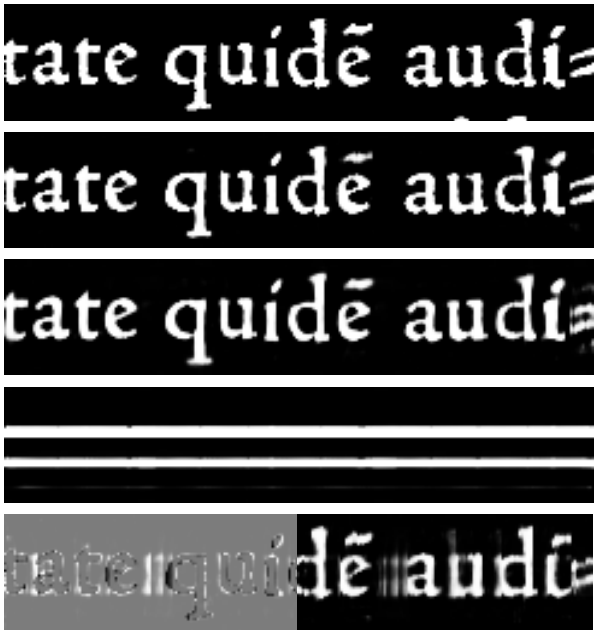

Figure 7.4: Comparison of input reconstructions The reconstruction after pre-training in “CTC-pretrain" (b) and fully training “CTC-multi" (c) is close to the original image (a). After fully training “CTC-pretrain", the network forgets the features learned for reconstruction and fails at the input reconstruction task (d). For the Seq2Seq architecture similarly, after pre-training both reconstruction and prediction are good, however after fully training “Seq2Seq-Latin+pretrain" only the prediction (e, right) is good.

Considering the two objectives of generating and aligning a sequence of discrete symbols and generating a sequence of continuous pixel values, this behaviour might not be surprising. In order to reconstruct the input image the network has to learn how to encode local variations in the pixel values. Instead the OCR objective requires condensed information about the local character shape and its sequential context. It can be concluded, that input reconstruction is not a suitable objective for any type of additional unsupervised training for CTC-based OCR, as it leads to bad parameter initializations and lowers OCR performance.

7.1.5 OCR with Seq2Seq

First the Seq2Seq model has been trained exactly as described in Wojna et al., 2017, which led to a quick convergence and a good training performance with a CER of 1.4. However, these results did not translate to unseen data. The described training parameters and changes managed to mitigate this over-fitting problem. The best result on the historical data set was achieved without pre-training and had a CER of 13.73. With pre-training the instability of the training increased, such that within 1000 training steps the CER could change by up to 30. This led to the pre-trained model training's divergence, but a positive effect of the pre-training in this scenario could be seen in the comparison of convergence speeds. Although pre-trained and baseline model describe roughly the same training curves, the pre-trained model started to converge much faster, which led to an offset between the curves that is about equal to the number of pre-training steps. Interestingly, the pre-trained encoder also retained its ability to predict how a text line image continues, but lost its ability to reconstruct it. Again the input reconstruction features were forgotten during the supervised training with the OCR objective.
On the alternative synthetic MNIST data, training was stable and training converged after about $3.000.000$ supervised training steps. The best result this time was achieved by a model with unsupervised pre-training with a CER of 13.93. Unlike the Latin data, the generated sequences were much shorter and completely random, without any sequential structure. Especially the much shorter input sequences seemed to have helped with the stability.

Model	CER
Seq2Seq-MNIST	23.9
Seq2Seq-MNIST+pretrain	13.93
Seq2Seq-Latin	13.73
Seq2Seq-Latin+pretrain	50.26 (div)

Table 7.3: OCR results for the Seq2Seq models. Pre-training showed a positive effect for training on the synthetic MNIST data, but the unstable training eventually diverged after applying the pre-trained model on the Latin data.

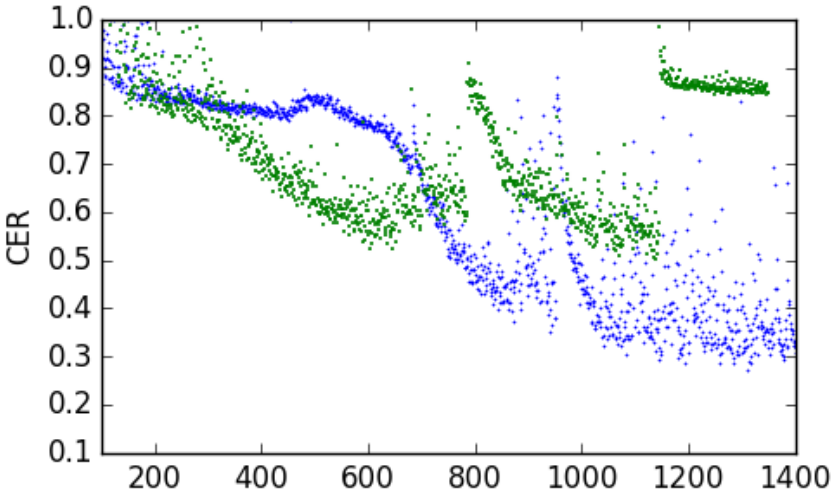

Figure 7.5: Training error of the Seq2Seq model on the Latin data The training progress is recorded every 1000 training steps for both Latin Seq2Seq models with (green) and without (blue) pre-training. The pre-trained model starts to converge faster and the delay roughly coincides with the length of pre-training period of 250.000 training steps. The instability of the training however leads to its divergence.

7.2 Adversarial Training with Cycle Consistency

The previous experiments highlighted the difficulty of applying any type of unsupervised pre-training to an LSTM-RNN based OCR approach and the need for novel approaches. This section gives an outlook on what the future of unsupervised training for OCR could look like by outlining a cycle-consistency based adversarial training approach using only mono-domain training data. The work described in this section is an outlook and includes assumptions and simplifications which might not necessarily be true for real world OCR applications. Addressing these restrictions will be left for future research.

Combining adversarial training and cycle consistency for unsupervised training using mono-domain data has recently been introduced for unsupervised translation tasks (Lample et al., 2017) and has since been extended to other fields such as speech recognition (Chung, Weng, Tong, and J. Glass, 2018). The original idea by Conneau et al., 2017 did not use adversarial training, but instead learned explicit word embeddings. These embeddings encoded

the statistical features of the underlying language and by aligning them along known or likely pivots, a Seq2Seq model would learn to map between the two domains without supervision. As pivots they used common words which in both languages had similar importance. This also meant, that the approach worked less well between fundamentally different languages that do not share such pivots. In OCR the existence of such pivots would be guaranteed, since both domains represent a single language and therefore share all language properties.
The work by Lample et al., 2017 further expanded on this approach through the use of adversarial training. Rather than explicitly learning an alignment of one embedding space to the other, a discriminators implicitly learned to align both embedding spaces with each other. This is further enforced by the use of a single generator for both languages that is trying to fool both discriminators, depending for which language it generates the output. Word embeddings were still used to map between the generator output and the respective languages.
In both scenarios cycle consistency was the key to make sure, that the learned mapping between the languages was not random. Both tasks had the objective to map every point of one latent space, to a point in the other. With adversarial training alone the model could learn to randomly map the input to any output in the other domain or it could learn to map all inputs to the same output. Cycle consistency enforces that the learned mapping is indeed a $1:1$ mapping (Almahairi et al., 2018).

Applying a similar approach to OCR seems straight forward, but comes with a number of problems. Learning word embeddings similar to the original approach would require segmented text lines, which are not available in most OCR tasks. OCR is also mostly performed on the character level, which makes word embeddings less useful, since wrong character predictions could lead to a non-existent embedding. Due to the different lengths of text and image sequence, there is also an additional requirement to learn an alignment between the input and output sequences. This alignment could be learned through an attention model, but the training instability and overall low performance in the last experiment made it apparent, that Seq2Seq models with attention are not the best fit for OCR. To avoid these problems the outlined approach uses a fixed size perceptive field for each recognized character.
The resulting approach is similar to the one proposed in Gupta et al., 2018b, both in concept and architecture, but differs in the use of cycle consistency as an additional training objective. Gupta et al., 2018b argued against the

use of cycle consistency, due to the added ambiguity it creates during the text rendering. In OCR there is only a single correct output sequence, but in the reverse direction, many different images can depict the same character sequences. While this is true, it also does not matter for the OCR model which text line in particular is generated, as long as the OCR engine can be used to turn it back into the same sequence. Instead, here it is argued that the cycle consistency constraint can help with the symbol grounding problem.

7.2.1 Architecture

In order to perform adversarial training between two domains, for either direction a generator as well as a discriminator is required. Since the goal is to also use cycle consistency as an additional source of information for the symbol grounding, this means the two generators should form a encoder-decoder architecture. For the generators a limited receptive field can be achieved best with a deep CNN architecture, which led to the choice of an architecture similar to a UNet (Ronneberger et al., 2015).
A UNet is an encoder-decoder architecture where encoder uses successive convolutions and down sampling to condense image information into a single feature vector, before the decoder up samples it into a target domain. Added skip connections between each encoder and decoder layer allow to simultaneously learn local features on multiple scales.

In the proposed architecture the encoder of the UNet is used as the OCR model, which condenses the image information into a text string. The decoding text renderer then expands this information back into a text line image. Since OCR is only a one-directional task, the skip connections between encoder and decoder are removed. Instead an LSTM-RNN layer is added via skip-connections before the final classification and output layer of each generator in order to better model the sequential nature of the application domain. This skip connection retains the fixed receptive field for each recognized character, while it enables the RNN to provide additional contextual information to the classifier.

Generators

Following the UNet architecture, the encoder consists of 8 convolutional layers with LeakyRelu activations (Maas et al., 2013). Also batch normalization has

been added to each layer to improve both training speed and stability (Ioffe and Szegedy, 2015). Unlike the original architecture the number of filters does not increase with each layer. Instead each layer uses a fixed number of n $3x3$ kernels, which should be sufficient given there is no need to model many high-frequency components. Also similar to the architecture introduced in Gupta et al., 2018b, after every 2 convolutions, the input dimensions are halved through max-pooling. Combined with padding in the convolution layer to keep the size fixed, this allows to adjust the receptive field for each predicted character according to the character size in the mono space font. The added RNN after the final convolution is a single BILSTM-RNN layer. The RNN output and the output of the final convolution layer are then combined into a fully connected Softmax layer for the classification.
Analogously the decoder uses the same basic architecture and consists of 8 convolutional layers with LeakyRelu activations and batch normalization. Instead of max pooling the decoder uses upsampling via transposed convolutional layers (Dumoulin and Visin, 2016) with $2x2$ filters to double the output dimensions. For symmetry these upsampling happens before rather than after each block of 2 convolutions. The final convolutional layer of the decoder is followed by a single layer BISLTM-RNN with a skip connection as well, that feeds into the linear output layer.

Discriminators

Building a discriminator that provides meaningful errors in OCR related fields comes with additional challenges (Yu et al., 2017). One problem in particular are the discrete target symbols. In adversarial training the generator has to make small adjustments according to the discriminators feedback. However, for discrete symbols these small adjustments will often be too small to result in a change of output symbols and the generator will stop learning. This can be prevented by learning a character embedding alongside the adversarial training (Press et al., 2017). If both the one-hot encoded real text sequence and the Softmax output sequence are mapped into a continuous embedding space, then discrimination can happen as normal. The same embedding is also used during the generator training. Additionally, label smoothing is applied to the one-hot encoded real sequences (Wojna et al., 2017).
With the generators' emphasis on local structure and limited perception, the discriminators have to enforce long range dependencies in the data. After early attempts with an RNN-based discriminator architecture the PatchGAN discriminator described in Gupta et al., 2018b has been chosen. It balances the size of the perceptive field with the number of parameters in the discriminator,

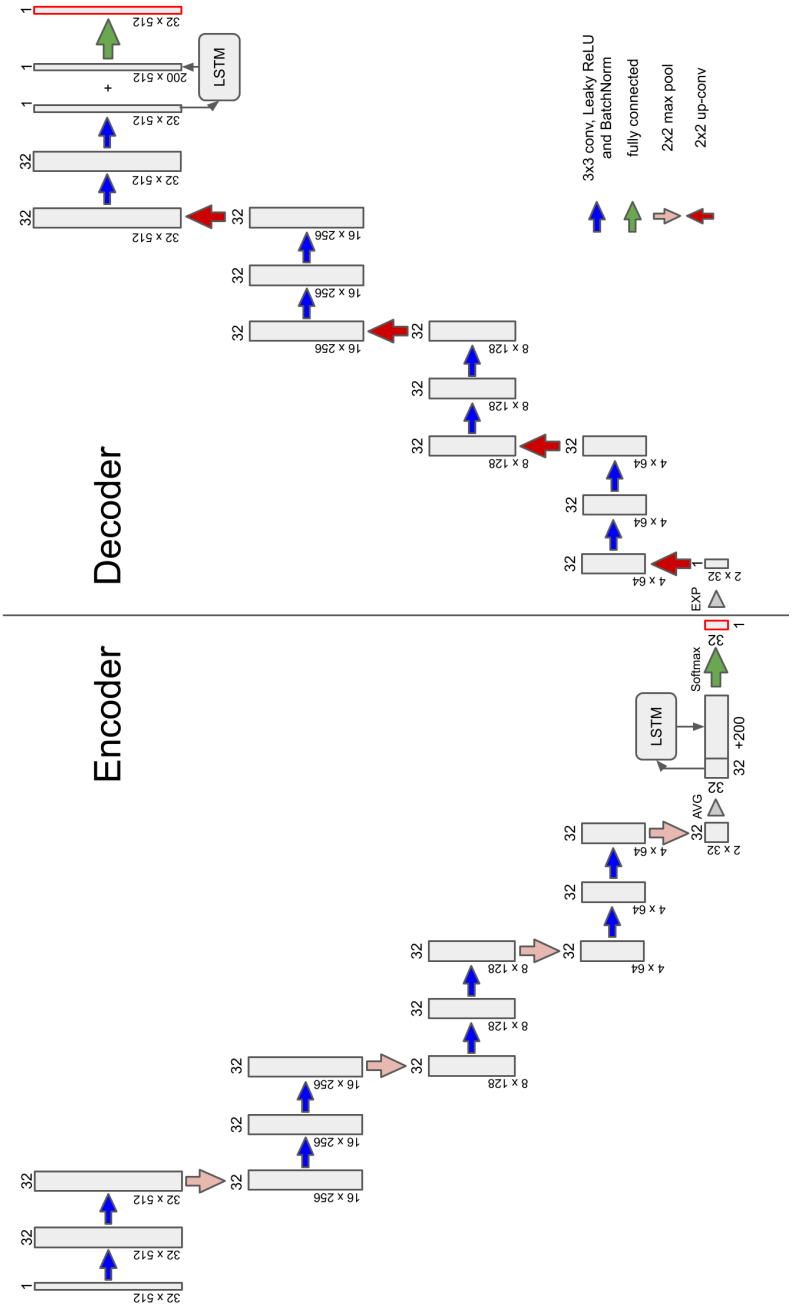

Figure 7.6: UNet-like architecture for both generators The generators in the text (left) and image domain (right) together form an UNet-like architecture with added LSTM-RNNs before the final layer. The output of the encoder is the OCRed text (bottom, red border), while the decoder produces a text line image (top, red border). Both generators consist of 8 convolutional layers and 4 max pooling or transposed convolutional layers respectively, followed by an LSTM-RNN before the fully connected output layer.

which improves training speed and performance. More importantly, it only sees partial sequences and therefore provides better error estimation for incomplete sequence predictions during the early training stages.
The used PatchGAN architecture for both discriminators is a 5 layer 1D-convolutional network with LeakyReLu activations and layer normalization (Ba et al., 2016) after every convolution. The 512 3xH filters in each layer effectively limit the receptive field to 21 characters, which is about $2/3$rd of the maximum 32 characters allowed. In the final layer the predictions over the patches are averaged into a single output for the whole sequence.

Restrictions

The outlined architecture is a proof of concept and as such introduces a couple of simplifications, which allow to focus on the unsupervised training capabilities. The main restriction is put on the input data, which is assumed to be in a mono space font. If each character takes roughly the same space, then the recognizer can use a fixed size receptive field to identify characters and align them with the output sequence.
Additional minor restrictions are imposed on the length of the input sequence and the size of the corresponding alphabets. This decreases the size of the resulting network architecture and lowers the computational complexity. Both restrictions are more out of convenience than necessity and can be removed more easily, but a larger network will also need longer training and more training data.

7.2.2 Training Objective

The training objective for this network is to minimize both the adversarial loss $L_{G,D}$ and cycle consistency loss L_{cyc}. The full loss is given as:

$$Loss_{total} = L_{cyc}^{txt} + L_{cyc}^{img} + \alpha(L_D^{txt} + L_G^{txt}) + \beta(L_D^{img} + L_G^{img}) \tag{7.3}$$

with α and β as scaling factors that control the relative strength of the different losses.
For the cycle consistency loss L_{cyc} multiple different options have been tried, both symmetrical, where the loss is the same in both domains, and asymmetrical, where a different loss is used in the text domain. Finally, the best results were achieved with a symmetric L_2 loss, between the original and reconstructed images as well as between the embedded input and reconstructed text sequence. Similarly for the adversarial loss, multiple options

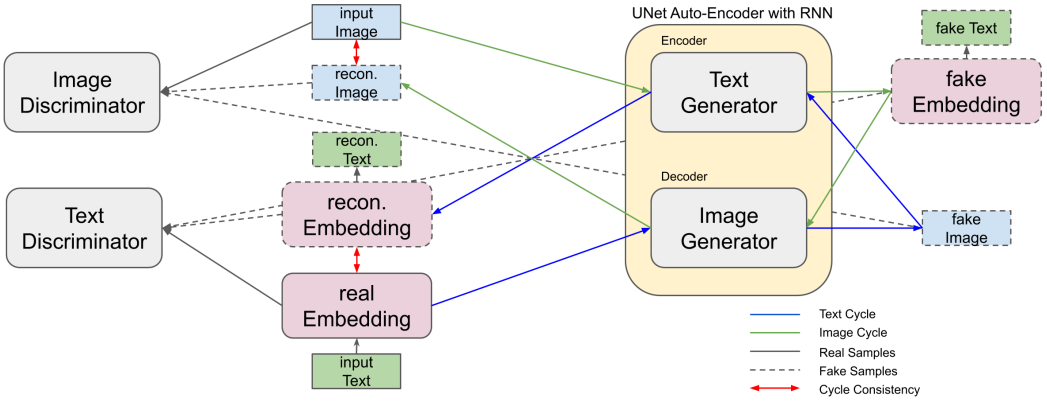

Figure 7.7: Schematic of adversarial training with cycle consistency. The non-paired image and text inputs perform two separate processing cycles (green and blue) through the architecture, resulting in a fake and a reconstructed output in each domain. These are used along side the original inputs for the adversarial training. All text inputs are processed in an embedding space, that turns the discrete symbols into continuous representations.

have been tested to mitigate the problems of adversarial training as discussed in the context of GANs in Section 3.2. The best result and most stable training could be achieved with the adversarial loss from the relativistic LSGAN (Jolicoeur-Martineau, 2018). It combines the balancing properties of the LSGAN with a conceptually different objective for the discriminator, which has been shown to prevent mode collapse. Rather than distinguishing real and fake by classifying them on either end of the spectrum, it is sufficient for the adversarial training, if the discriminator identifies a fake input as being less likely to be real than a random real input. Vice versa the generator has the opposite objective to fool the discriminator into identifying its output as more likely to be real, than a random real input. This can be formalized into a loss as

$$L_D = \mathbb{E}_{x_{real}, x_{fake}}[L_2(D(x_{real}) - D(x_{fake}))] \tag{7.4}$$

$$L_G = \mathbb{E}_{x_{real}, x_{fake}}[L_2(D(x_{fake}) - D(x_{real}))] \tag{7.5}$$

Here x_{real} and x_{fake} denote the real and fake inputs to the discriminator, either from the training data or generated by the generator. This includes the fake x_{recon} sample that are generated as part of cycle consistency objective. In practice, the base probability for a random real input to be real or a random fake input to be fake, is sampled from the batch statistics at training time.

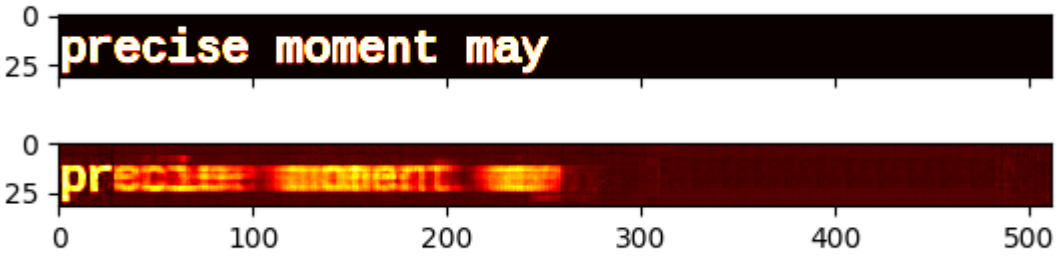

Figure 7.8: Cycle consistency results in the image domain Even after training to convergence the cycle consistency objective in the image domain is not completely fulfilled.

7.2.3 Experimental Setup

The model architecture restricted the type of training data to be in a mono-space font, so for the purpose of evaluating the performance the English Gigaword data set (Graff et al., 2003) has been used to render synthetic text line images with a mono-space font. Furthermore, the alphabet is also limited, so text lines with anything but lower case characters have been ignored. For the training data 10.000 text line images have been generated with an additional 1.000 for validation and test set each. The fixed size input of he CNN-based architecture limits the input to $32x512$ and $32x27$ for the text generator and the image generator respectively. Text lines with more than the allowed 32 characters were split into multiple text lines with about the same length before rendering. Finally, the number of filters n for each convolutional layer in the generators has been chosen to be 32.
The training was performed with an RMSprop optimizer, a learning rate of $1e-3$, a batch size of 64 and was continued for about 30 million training steps. Each batch was independently sampled from the text and image data, which resulted in unpaired training samples. During adversarial training all the generator outputs, including the ones from the cycle consistency step, have been used as fake samples. The scaling parameters in the loss function have been chosen as $\alpha = 10.0$ and $\beta = 10.0$.
The evaluation of the trained model is performed by calculating the CER using the Levenshtein distance.

7.2.4 Results

During the training process relatively fast improvements could be observed in the quality of the OCR predictions. The generalization onto the validation and test set though lags behind and never reaches the training performance, which indicates some level of over-fitting. After about $20000k$ training steps the model reaches very good OCR performance on the training data with a CER less than 1.00, but still produces more than 10 times the errors on the test set. Further, training first leads to a relapse, where performance temporarily worsened, but ultimately the best model is recorded after about $30000k$ iterations with a test CER of 7.78. Continuing training did not improve the test performance any further and instead lead to another relapse with worse predictions.
While the OCR prediction and the text reconstruction quickly show training progress, the image generation and reconstruction tasks improve much more slowly and never succeeded in producing full realistic text line images. Lowering the relative scaling of the adversarial training in the text domain by reducing α does improve the image reconstruction and generation qualities of the architecture, but it also drastically reduces the quality of the OCR output. This raises the question whether the cycle consistency objective and the added image generation tasks are at all helpful to the whole architecture. An additional experiment where the cycle consistency objective was removed, shows that it is essential for the generator to produce meaningful OCR output.

N_{train}	OCR	TRAIN-CER	TEST-CER
	that consultation is		
5000k	"th t erohe tatien i"	28.94	42.37
10000k	"that cerselttien e"	8.84	21.94
15000k	"that corsultation is"	1.50	12.56
20000k	"ttat comaultation ie"	0.76	11.84
25000k	"ttat corseltation ie"	1.16	13.92
30000k	"that consultatior is"	0.50	7.78

Table 7.4: OCR prediction progress, test and train error for the unsupervised OCR training The reduced test performance indicates towards some amount of over-fitting. The last reported model shows the best results

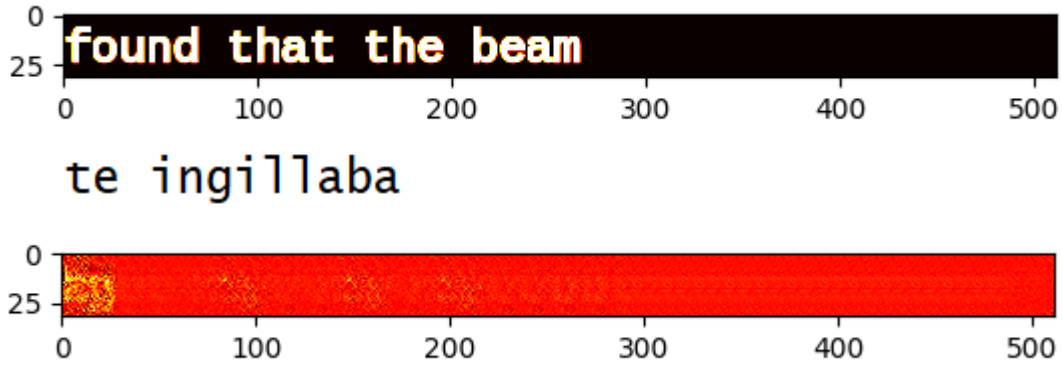

Figure 7.9: Effect of no cycle consistency objective After training without the cycle consistency objective, the input image (top) can not be reconstructed (bottom) and the OCR output (middle) does not represent the displayed text.

7.3 Summary

In the first part of this chapter the goal was to explore how the standard LSTM-RNN based OCR approach can be improved through the use of unlabeled training data or more specifically through unsupervised training with an input reconstruction objective. Inspired by the work of the previous chapter, multiple training schemes using an encoder-decoder architecture were introduced. In all cases the LSTM-RNN based OCR model was used as an encoder and combined with a decoder that has a similar architecture. Additionally, a training scheme based on a Seq2Seq model was tested, a popular RNN architecture from other language processing domains that has shown to profit from training with additional unlabeled data. The performance was evaluated using a historical data set, a prime application domain for such training methods.
In all but one scenario the baseline model trained with only supervised training showed the best performance, with unsupervised pre-training via input reconstruction coming in second. The exception was a Seq2Seq model that was trained on synthetic data. In absolute numbers however, the Seq2Seq architecture had a much lower performance, needed longer training and showed instabilities during training.

From the results it can be concluded, that input reconstruction in the image domain is not a good objective for training LSTM-RNN based OCR in an unsupervised manner. There is no indication for any synergy with supervised

training via CTC loss. Instead, the unsupervised training led to a reduced performance and bad parameter initializations. Future research could focus on finding a different objective in the text domain, so there is a higher chance for synergistic effects with the supervised training.

In the second part of the chapter the focus was shifted towards the future of unsupervised OCR training. For this purpose an approach using unpaired mono-domain training data has been outlined. Similar to GANs the approach makes use of adversarial training combined with a cycle consistency constraint. This resulted in a UNet-like auto-encoder architecture as generators for both domains and two fully convolutional discriminators. A limited amount of sequential learning capabilities were introduced by adding LSTM-RNNs to the otherwise fully convolutional encoder and decoder. This is not sufficient to overrule the fixed size receptive fields of the encoder, as indicated by early attempts with non mono-space font texts such as the historical data introduced in 4.4.3, but allows the network to enrich the recognition with information about the local context and the sequence structure.

The outlined approach did not achieve the performance shown in similar work by Gupta et al., 2018b, but it was also not possible to reproduce their described approach. The results shown here as well as the collected experience during the experiments highlighted a need for cycle consistency as an additional restriction for the symbol grounding. They also show the possibility of unsupervised OCR training with deep learning and encourage further research into the field. Such research will be necessary to address the open challenges and current limitations. It is also unlikely that the outlined approach, architecture and parameters already exhaust the full potential of the outlined approach and further optimizations will be necessary for future development.

CHAPTER 8

SUMMARY

This thesis introduced multiple approaches on how state-of-the-art LSTM-RNN based OCR models can be trained in the absence of transcribed training data. While it is the most common type of application scenario, not much work on the topic had been done previous to this work. Where possible approaches, concepts and architectures have been validated through experiments with historical documents, an application domain with mostly unsupervised training case.

In Chapter 5 the concept of training LSTM-RNN with erroneous ground truth data has been introduced and applied to OCR. After showing the effectiveness of the concept on transcriptions with multiple types of artificially introduced errors, the concept has been applied to a historical document. The proposed processing pipeline uses segmentation, clustering and limited manual labeling to generate synthetic transcriptions and uses them for supervised LSTM-RNN training. The results showed an OCR performance on par with the baseline model, that was trained on expert transcriptions.
Additionally, layman annotations were proposed as another possible source for cheap erroneous transcriptions. To close the gap to expert annotations and mitigate annotation errors due to unrecognizable characters, a new concept for uncertain transcriptions has been introduced. The resulting fuzzy ground truth can be used to train standard LSTM-RNN based OCR models through training with a modified CTC loss, that allows for such uncertain transcriptions. It could be shown that fuzzy ground truth is better than guessing and a training scheme that combines it with the proposed processing pipeline.

In Chapter 6 a solution to the open problem of transcription free model evaluation has been proposed. The proposed concept uses input reconstruction in the image domain and shows how OCR errors are linked to various types of image reconstruction errors . The concept was realised through a LSTM-RNN based text line generator, that combined with an LSTM-RNN based OCR model forms an encoder-decoder architecture, which allows for input reconstruction in the image domain. Although there are multiple conceptual differences to the standard evaluation, experimentation on historical data showed, that both methods perform similar in the task of model selection.

In Chapter 7 first the use of unlabeled training data through use of input reconstruction was explored, inspired by the encoder-decoder architecture introduced in Chapter 6. The evaluation on historical data showed no synergistic effect between the input reconstruction and the CTC objective as all attempts of additional unsupervised training were unsuccessful in improving the final baseline performance. Instead, it lead to conflicting parameter updates and bad initialization.
Finally an outline of a possible future of unsupervised OCR is given by introducing a cycle consistent adversarial training approach for LSTM-RNN. The unsupervised training currently imposes strict limitations on possible application scenarios, but it showed promising OCR results and can serve as a basis for future research in the field.

The following Section 8.1 provides an overview of some observed limitations and gives somes concluding remarks in the context of the main goal of this thesis and the formulated research hypotheses. Afterwards an outlook of the future of OCR is given and some related possible future work is outlined.

8.1 Conclusion

The goal of this thesis was to find ways to use sequence learning for the training of state-of-the-art LSTM models in unsupervised training cases. Three hypothesis have been formulated to answer this question (see Section 1.2).

The work presented in Chapter 5 validated the first hypothesis **H1**. The effectiveness of training LSTM-RNNs for sequence labeling with erroneous transcriptions has been shown in multiple experiments and it could repeatedly outperform the baseline as well as produce competitive results compared to training with expert transcriptions. Furthermore, using an OCR model with

no other components has confirmed that this is indeed a property of the LSTM-RNN. It has also been showcased how to practically apply the concept in an unsupervised training case using historical document.
On the other hand minor objectives related to the name of the "anyOCR" pipeline could not be fully satisfied. Although the introduced pipeline has conceptually few limitations, in practice the implemented approaches for segmentation and clustering are a limiting factor for applications on more complex languages like Chinese or Arabic. The same is true for the accessibility. Both the proposed pipeline as well as the introduction of fuzzy ground truth theoretically lower the need for language specific expertise, but the experiments indicated that annotators even with just basic knowledge about the target language performed better and faster.

In Chapter 6 the second hypothesis **H2** has been validated. The introduced encoder-decoder architecture allowed applying input reconstruction as an evaluation objective for the performance of the OCR encoder. Errors in the OCR could be measured in the image domain by linking them to image reconstruction errors and experiments on a historical document data set have shown that this method can be used for model selection purposes through a relative model ranking.
The results are similar to those of the standard OCR evaluation in the text domain, but do not fully agree. However, in the discussion of the conceptual differences it has been highlighted that some of these differences are actually favorable to evaluating OCR models for generalization.

Finally, in Chapter 7 the last hypothesis **H3** could be confirmed. The outlined training process combined adversarial training with a cycle consistency constraint, which successfully enabled unsupervised training for LSTM-RNN based OCR. More specifically, the discriminators learned to enforce the properties of their mono-domain reference data onto the generators, which learned to align them with each other through the cycle consistency objective. From the experiences collected during various experiments, it became clear that cycle consistency is a necessary component for this type of unsupervised learning.
While these are encouraging results for further research, they are also enabled by strict limitations on the approach. One of the advantages of LSTM-RNN based OCR models is their ability to learn an alignment between input and output sequence, which could not be realised in this work and had to be enforced through the data. Furthermore, the proposed architecture could not achieve the performance shown in similar work by Gupta et al., 2018b, but neither was it possible to reproduce their results either.

8.2 Outlook and Future Work

Before highlighting some possible future work on the contributions presented in this thesis, a brief outlook is given on the future of OCR in general, with a focus on applications to unsupervised training cases.

Recent developments in OCR have shown a pivot towards more end-to-end training (Clanuwat et al., 2019; Tensmeyer and Wigington, 2019). This is likely to produce novel approaches with less and less restrictions on the corresponding transcription as well as the OCR engineer's ability to build optimized processing pipelines. Instead, the current need of text line level transcriptions will move to page level or document level. As part of this development OCR on complex layouts will also see a significant improvement.
A different avenue in OCR research is the creation of synthetic image data using generative approaches (Pondenkandath et al., 2019), which will enable the use of synthetic data more broadly and will give a rise to data augmentation techniques, further reducing the need for transcriptions.
A third avenue for the development of OCR is outlined by the work in Chapter 7 and Gupta et al., 2018b. If this work can be extended to be generally applicable, then unsupervised training on mono domain data is likely to become the new standard for OCR. However, text domain data is still required so some level of transcription will still be needed.

For the proposed methods in Chapter 5 this also means, that they have possible future applications. While the specific implementation of the "anyOCR" pipeline might not be the future of OCR, conceptually it is not limited in the same way. Instead, the erroneous transcription data could come from other sources, including one of the mentioned avenues. This way it can serve as a post-processing error correction for other OCR approaches. Alternatively the high training performance observed on the unsupervised training approach could serve as an alignment tool to generated paired training data from mono-domain inputs. It would also be possible to combine multiple such alignments through fuzzy ground truth for further improvements.
The proposed evaluation method from Chapter 6 is similarly independent of the exact OCR architecture, training approach or data availability and could be applied to any of the mentioned approaches.

With this outlook in mind, the following highlights some more detailed possibilities for future work:

- **anyOCR pipeline** The current implementations for both the segmentation and the clustering impose direct or indirect limitations on possible application scenarios of the pipeline. This includes languages such as Arabic with highly connected scripts or Chinese with a large character pool. The first calls for further research into robust segmentation techniques, such as the exploration of deep learning approaches by Saha, 2019.
  For the latter an alternative clustering and annotation scheme will be necessary, since the large number of different character makes both impractical.

- **fuzzy ground truth** The introduced concept of fuzzy ground truth allows for uncertainty in the transcriptions. The modified CTC-based training allows LSTM-RNN to learn from these uncertainties. In its current form the uncertainties however are limited to the same length. Further modifications to the CTC-based training are required to also allow for different length uncertainties, which are not uncommon in historical data sets.

- **text line reconstruction** In the transcription-free model evaluation it was noted that the quality of the reconstruction introduce a base error that is unrelated to the OCR performance. Future research should therefore focus on how to improve the overall reconstruction accuracy, especially in terms of sharpness. An attempt by Sinha et al., 2019 with a generative approach has shown the difficulty of this problem.

- **unsupervised pre-training** During the exploration of unsupervised pre-training for LSTM-RNN training with CTC it became apparent that input reconstruction on the image domain did not provide a good initialization for the OCR model. Instead it could be useful to research unsupervised pre-training in the text domain. With the final goal of producing text output, such a pre-training could help to improve the inherent language modeling abilities and strengthen the sequential processing capabilities.

- **adversarial training** The cycle consistent adversarial training approach for unsupervised OCR training has been introduced as a proof of concept and although it has shown promising early results, further research is necessary to verify that this approach is also applicable to real OCR applications. This especially means to address the current restrictions on the input data. Further systematic studies are also needed to quantify the effect of the scaling between the errors and the benefit of each component of the training process.

BIBLIOGRAPHY

Abtahi, F., Z. Zhu, A. M. Burry (May 2015). “A deep reinforcement learning approach to character segmentation of license plate images.” In: *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*. ISSN: null, pp. 539–542.

Afzal, M. Z., M. Krämer, S. S. Bukhari, M. R. Yousefi, F. Shafait, T. M. Breuel (2013). “Robust binarization of stereo and monocular document images using percentile filter.” In: *International Workshop on Camera-Based Document Analysis and Recognition*. Springer, pp. 139–149.

Aghabozorgi, S., A. Seyed Shirkhorshidi, T. Ying Wah (Oct. 2015). “Time-series clustering – A decade review.” en. In: *Information Systems* 53, pp. 16–38.

Ahmad, R., M. Z. Afzal, S. F. Rashid, M. Liwicki, T. Breuel (2015). “Scale and rotation invariant OCR for Pashto cursive script using MDLSTM network.” In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 1101–1105.

Almahairi, A., S. Rajeswar, A. Sordoni, P. Bachman, A. Courville (2018). “Augmented cyclegan: Learning many-to-many mappings from unpaired data.” In: *arXiv preprint arXiv:1802.10151*.

Amornkosit, T. (2018). “Applying User-Centered Design principles to improve user experience of anyOCRWeb and anyTrain web applications.” MA thesis. TU Kaiserslautern, Germany.

Arjovsky, M., S. Chintala, L. Bottou (2017). “Wasserstein gan.” In: *arXiv preprint arXiv:1701.07875*.

Ba, J. L., J. R. Kiros, G. E. Hinton (2016). “Layer normalization.” In: *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, Y. Bengio (Mar. 2017). “An Actor-Critic Algorithm for Sequence Prediction.” In: *arXiv:1607.07086 [cs]*. arXiv: 1607.07086.

Bahdanau, D., K. Cho, Y. Bengio (2014). “Neural machine translation by jointly learning to align and translate.” In: *arXiv preprint arXiv:1409.0473*.

Bao, W., J. Yue, Y. Rao (July 2017). “A deep learning framework for financial time series using stacked autoencoders and long-short term memory.” en. In: *PLOS ONE* 12.7, e0180944.

Belongie, S., J. Malik, J. Puzicha (2002). “Shape matching and object recognition using shape contexts.” In: *IEEE transactions on pattern analysis and machine intelligence* 24.4, pp. 509–522.

Bengio, Y., P. Lamblin, D. Popovici, H. Larochelle (2006). “Greedy Layer-wise Training of Deep Networks.” In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. NIPS’06. Canada: MIT Press, pp. 153–160.

Berg-Kirkpatrick, T., G. Durrett, D. Klein (2013). “Unsupervised transcription of historical documents.” In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 207–217.

Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. (2016). “End to end learning for self-driving cars.” In: *arXiv preprint arXiv:1604.07316*.

Breuel, T. M. (2008). “The OCRopus open source OCR system.” In: *Document Recognition and Retrieval XV*. Vol. 6815. International Society for Optics and Photonics, 68150F.

Breuel, T. M. (2017). “High performance text recognition using a hybrid convolutional-lstm implementation.” In: *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*. Vol. 1. IEEE, pp. 11–16.

Breuel, T. M., A. Ul-Hasan, M. A. Al-Azawi, F. Shafait (2013). “High-performance OCR for printed English and Fraktur using LSTM networks.” In: *2013 12th International Conference on Document Analysis and Recognition*. IEEE, pp. 683–687.

Breuel, T. M. (Aug. 2015). “Benchmarking of LSTM Networks.” en. In: *arXiv:1508.02774 [cs]*. arXiv: 1508.02774.

Buckland, M. K. (1997). “What is a “document”?” In: *Journal of the American society for information science* 48.9, pp. 804–809.

Bukhari, S. S., A. Kadi, M. A. Jouneh, F. M. Mir, A. Dengel (2017). “anyocr: An open-source ocr system for historical archives.” In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. IEEE, pp. 305–310.

Casey, R. G., E. Lecolinet (1996). “A survey of methods and strategies in character segmentation.” In: *IEEE transactions on pattern analysis and machine intelligence* 18.7, pp. 690–706.

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio (Sept. 2014). “Learning Phrase Representations

using RNN Encoder-Decoder for Statistical Machine Translation." In: *arXiv:1406.1078 [cs, stat]*. arXiv: 1406.1078.

Chung, Y.-A., W.-H. Weng, S. Tong, J. Glass (2018). "Unsupervised cross-modal alignment of speech and text embedding spaces." In: *Advances in Neural Information Processing Systems*, pp. 7354–7364.

Chung, Y.-A., W.-H. Weng, S. Tong, J. R. Glass (2018). "Towards Unsupervised Speech-to-text Translation." In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7170–7174.

Clanuwat, T., A. Lamb, A. Kitamoto (2019). "KuroNet: Pre-Modern Japanese Kuzushiji Character Recognition with Deep Learning." In: *arXiv preprint arXiv:1910.09433*.

Conneau, A., G. Lample, M. Ranzato, L. Denoyer, H. Jégou (2017). "Word translation without parallel data." In: *arXiv preprint arXiv:1710.04087*.

Doermann, D., K. Tombre, et al. (2014). *Handbook of document image processing and recognition*. Springer.

Dumoulin, V., F. Visin (2016). "A guide to convolution arithmetic for deep learning." In: *arXiv preprint arXiv:1603.07285*.

Elman, J. L. (1990). "Finding structure in time." In: *Cognitive science* 14.2, pp. 179–211.

Fedus, W., I. Goodfellow, A. M. Dai (Mar. 2018). "MaskGAN: Better Text Generation via Filling in the_____." en. In: *arXiv:1801.07736 [cs, stat]*. arXiv: 1801.07736.

Gers, F. A., N. N. Schraudolph, J. Schmidhuber (2002). "Learning Precise Timing with LSTM Recurrent Networks." en. In: p. 29.

Gers, F. A., J. Schmidhuber, F. Cummins (Oct. 2000). "Learning to Forget: Continual Prediction with LSTM." en. In: *Neural Computation* 12.10, pp. 2451–2471.

Goodfellow, I. (2016). "NIPS 2016 tutorial: Generative adversarial networks." In: *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., Y. Bengio, A. Courville (2016). *Deep learning*. MIT press.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio (2014). "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger. Curran Associates, Inc., pp. 2672–2680.

Graff, D., J. Kong, K. Chen, K. Maeda (2003). "English gigaword." In: *Linguistic Data Consortium, Philadelphia* 4.1, p. 34.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. en. Vol. 385. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg.

Graves, A. (2013). “Generating sequences with recurrent neural networks.” In: *arXiv preprint arXiv:1308.0850*.

Graves, A. (June 2014). “Generating Sequences With Recurrent Neural Networks.” In: *arXiv:1308.0850 [cs]*. arXiv: 1308.0850.

Graves, A., S. Fernández, J. Schmidhuber (2005). “Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition.” In: *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*. ICANN’05. event-place: Warsaw, Poland. Berlin, Heidelberg: Springer-Verlag, pp. 799–804.

Graves, A., M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber (2008). “A novel connectionist system for unconstrained handwriting recognition.” In: *IEEE transactions on pattern analysis and machine intelligence* 31.5, pp. 855–868.

Graves, A., A.-r. Mohamed, G. Hinton (2013). “Speech recognition with deep recurrent neural networks.” In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 6645–6649.

Graves, A., J. Schmidhuber (2009). “Offline handwriting recognition with multidimensional recurrent neural networks.” In: *Advances in neural information processing systems*, pp. 545–552.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville (Dec. 2017). “Improved Training of Wasserstein GANs.” en. In: *arXiv:1704.00028 [cs, stat]*. arXiv: 1704.00028.

Gupta, A., A. Vedaldi, A. Zisserman (2016). “Synthetic data for text localisation in natural images.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2315–2324.

Gupta, A., A. Vedaldi, A. Zisserman (2018a). “Learning to Read by Spelling.” en. In: p. 15.

Gupta, A., A. Vedaldi, A. Zisserman (2018b). “Learning to read by spelling: Towards unsupervised text recognition.” In: *arXiv preprint arXiv:1809.08675*.

Harnad, S. (June 1990). “The symbol grounding problem.” en. In: *Physica D: Nonlinear Phenomena* 42.1, pp. 335–346.

Ul-Hasan, A., S. B. Ahmed, F. Rashid, F. Shafait, T. M. Breuel (2013). “Offline printed Urdu Nastaleeq script recognition with bidirectional LSTM networks.” In: *2013 12th International Conference on Document Analysis and Recognition*. IEEE, pp. 1061–1065.

Ul-Hasan, A., T. M. Breuel (2013). “Can we build language-independent OCR using LSTM networks?” In: *Proceedings of the 4th International Workshop on Multilingual OCR*, pp. 1–5.

Ul-Hasan, A., S. S. Bukhari, A. Dengel (2016). “Ocroract: A sequence learning ocr system trained on isolated characters.” In: *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE, pp. 174–179.

Ho, T. K., G. Nagy (2000). “OCR with no shape training.” In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 4. IEEE, pp. 27–30.

Hochreiter, S., J. Schmidhuber (1997). “Long short-term memory.” In: *Neural computation* 9.8, pp. 1735–1780.

Hoff, K. J., S. Lange, A. Lomsadze, M. Borodovsky, M. Stanke (Mar. 2016). “BRAKER1: Unsupervised RNA-Seq-Based Genome Annotation with GeneMark-ET and AUGUSTUS.” en. In: *Bioinformatics* 32.5, pp. 767–769.

Hopfield, J. J. (1982). “Neural networks and physical systems with emergent collective computational abilities.” In: *Proceedings of the national academy of sciences* 79.8, pp. 2554–2558.

Huang, C.-Z. A., A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, D. Eck (Dec. 2018). “Music Transformer.” In: *arXiv:1809.04281 [cs, eess, stat]*. arXiv: 1809.04281.

Huang, G., E. Learned-Miller, A. McCallum (2007). “Cryptogram decoding for optical character recognition.” In: *International Conference on Document Analysis and Recognition*.

Inoue, M., T. Yamashita, T. Nishida (2019). “Robot Path Planning by LSTM Network Under Changing Environment.” en. In: *Advances in Computer Communication and Computational Sciences*. Ed. by S. K. Bhatia, S. Tiwari, K. K. Mishra, M. C. Trivedi. Advances in Intelligent Systems and Computing. Singapore: Springer, pp. 317–329.

Ioffe, S., C. Szegedy (2015). “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In: *arXiv preprint arXiv:1502.03167*.

Jaderberg, M., K. Simonyan, A. Vedaldi, A. Zisserman (2014). “Synthetic data and artificial neural networks for natural scene text recognition.” In: *arXiv preprint arXiv:1406.2227*.

Jaeger, H., H. Haas (2004). “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication.” In: *science* 304.5667, pp. 78–80.

Jenckel, M., S. S. Bukhari, A. Dengel (2016a). “anyOCR: A sequence learning based OCR system for unlabeled historical documents.” In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2016 23rd International Conference on Pattern Recognition (ICPR). Cancun, Mexico, pp. 4035–4040.

Jenckel, M., S. S. Bukhari, A. Dengel (2016b). “Clustering Benchmark for Characters in Historical Documents.” In: *12th IAPR International Workshop on Document Analysis Systems DAS 2016*. Santorini, Greece, p. 2.

Jenckel, M., S. S. Bukhari, A. Dengel (2017). “Training LSTM-RNN with Imperfect Transcription: Limitations and Outcomes.” In: *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing*. HIP2017. event-place: Kyoto, Japan. Kyoto, Japan, pp. 48–53.

Jenckel, M., S. S. Bukhari, A. Dengel (2018). “Transcription Free LSTM OCR Model Evaluation.” In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). Niagara Falls, USA, pp. 122–126.

Jenckel, M., S. S. Bukhari, A. Dengel (2019). “Analysis of Unsupervised Training Approaches for LSTM-based OCR.” In: *Proceedings of the 15h International Conference on Document Analysis and Recognition (ICDAR)*. 2019 15h International Conference on Document Analysis and Recognition (ICDAR). Sydney, Australia.

Jenckel, M., S. S. Parkala, S. S. Bukhari, A. Dengel (2018). “Impact of Training LSTM-RNN with Fuzzy Ground Truth:” in: *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods*. 7th International Conference on Pattern Recognition Applications and Methods. Funchal, Madeira, Portugal, pp. 388–393.

Jolicoeur-Martineau, A. (2018). “The relativistic discriminator: a key element missing from standard GAN.” In: *arXiv preprint arXiv:1807.00734*.

Jordan, M. I. (1997). “Serial order: A parallel distributed processing approach.” In: *Advances in psychology*. Vol. 121. Elsevier, pp. 471–495.

Junaidi, A., G. A. Fink, et al. (2011). “A semi-supervised ensemble learning approach for character labeling with minimal human effort.” In: *2011 International Conference on Document Analysis and Recognition*. IEEE, pp. 259–263.

Kahle, P., S. Colutto, G. Hackl, G. Mühlberger (2017). “Transkribus-A service platform for transcription, recognition and retrieval of historical documents.” In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 4. IEEE, pp. 19–24.

Khan, R. (2018). “anyTrain: A Web Application for Training anyOCR Model for Any Language and Script.” MA thesis. TU Kaiserslautern, Germany.

Kidd, E., S. Donnelly, M. H. Christiansen (Feb. 2018). “Individual Differences in Language Acquisition and Processing.” en. In: *Trends in Cognitive Sciences* 22.2, pp. 154–169.

Krizhevsky, A., I. Sutskever, G. E. Hinton (2012). “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems*, pp. 1097–1105.

Lample, G., A. Conneau, L. Denoyer, M. Ranzato (2017). "Unsupervised machine translation using monolingual corpora only." In: *arXiv preprint arXiv:1711.00043*.

Larsson, G., M. Maire, G. Shakhnarovich (2016). "Learning representations for automatic colorization." In: *European Conference on Computer Vision*. Springer, pp. 577–593.

LeCun, Y., L. Bottou, Y. Bengio, P. Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lee, C.-Y., S. Osindero (June 2016). "Recursive Recurrent Nets with Attention Modeling for OCR in the Wild." en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, pp. 2231–2239.

Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals." In: *Soviet physics doklady*. Vol. 10. 8, pp. 707–710.

Lloyd, S. (1982). "Least squares quantization in PCM." In: *IEEE transactions on information theory* 28.2, pp. 129–137.

Lotter, W., G. Kreiman, D. Cox (2016). "Deep predictive coding networks for video prediction and unsupervised learning." In: *arXiv preprint arXiv:1605.08104*.

Louloudis, G., B. Gatos, I. Pratikakis, C. Halatsis (Dec. 2009). "Text line and word segmentation of handwritten documents." en. In: *Pattern Recognition* 42.12, pp. 3169–3183.

Lv, Y., Y. Duan, W. Kang, Z. Li, F.-Y. Wang (2014). "Traffic flow prediction with big data: a deep learning approach." In: *IEEE Transactions on Intelligent Transportation Systems* 16.2, pp. 865–873.

Maas, A. L., A. Y. Hannun, A. Y. Ng (2013). "Rectifier nonlinearities improve neural network acoustic models." In: *Proc. icml*. Vol. 30. 1, p. 3.

Mao, X., Q. Li, H. Xie, R. Y. Lau, Z. Wang, S. Paul Smolley (2017). "Least squares generative adversarial networks." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802.

McCulloch, W. S., W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis (Feb. 2015). "Human-level control through deep reinforcement learning." en. In: *Nature* 518.7540, pp. 529–533.

Moysset, B., R. Messina (2019). “Are 2D-LSTM really dead for offline text recognition?” In: *International Journal on Document Analysis and Recognition (IJDAR)* 22.3, pp. 193–208.

Noroozi, M., P. Favaro (Mar. 2016). “Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles.” In:

Oord, A. v. d., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu (Sept. 2016). “WaveNet: A Generative Model for Raw Audio.” In: *arXiv:1609.03499 [cs]*. arXiv: 1609.03499.

Oord, A. v. d., Y. Li, O. Vinyals (2018). “Representation learning with contrastive predictive coding.” In: *arXiv preprint arXiv:1807.03748*.

Papavassiliou, V., T. Stafylakis, V. Katsouros, G. Carayannis (Jan. 2010). “Handwritten document image segmentation into text lines and words.” en. In: *Pattern Recognition* 43.1, pp. 369–377.

Papineni, K., S. Roukos, T. Ward, W.-J. Zhu (2002). “BLEU: a method for automatic evaluation of machine translation.” In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 311–318.

Pascanu, R., T. Mikolov, Y. Bengio (2013). “On the difficulty of training recurrent neural networks.” In: *International Conference on Machine Learning*, pp. 1310–1318.

Pondenkandath, V., M. Alberti, M. Diatta, R. Ingold, M. Liwicki (2019). “Historical Document Synthesis with Generative Adversarial Networks.” In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Vol. 5. IEEE, pp. 146–151.

Press, O., A. Bar, B. Bogin, J. Berant, L. Wolf (2017). “Language generation with recurrent generative adversarial networks without pre-training.” In: *arXiv preprint arXiv:1706.01399*.

Qin, S., R. Manduchi (Nov. 2017). “Cascaded Segmentation-Detection Networks for Word-Level Text Spotting.” In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. ISSN: 2379-2140, pp. 1275–1282.

Ramachandran, P., P. J. Liu, Q. V. Le (2016). “Unsupervised pretraining for sequence to sequence learning.” In: *arXiv preprint arXiv:1611.02683*.

Raue, F., A. Dengel, T. M. Breuel, M. Liwicki (Jan. 2018). “Symbol Grounding Association in Multimodal Sequences with Missing Elements.” In: *J. Artif. Int. Res.* 61.1, pp. 787–806.

Ravi, D., C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, G.-Z. Yang (2016). “Deep learning for health informatics.” In: *IEEE journal of biomedical and health informatics* 21.1, pp. 4–21.

Renals, S., N. Morgan, H. Bourlard, M. Cohen, H. Franco (1994). "Connectionist probability estimators in HMM speech recognition." In: *IEEE Transactions on Speech and Audio Processing* 2.1, pp. 161–174.

Retsinas, G., B. Gatos, A. Antonacopoulos, G. Louloudis, N. Stamatopoulos (2015). "Historical typewritten document recognition using minimal user interaction." In: *Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing*, pp. 31–38.

Reul, C., C. Wick, U. Springmann, F. Puppe (2017). "Transfer learning for OCRopus model training on early printed books." In: *arXiv preprint arXiv:1712.05586*.

Ronneberger, O., P. Fischer, T. Brox (2015). "U-net: Convolutional networks for biomedical image segmentation." In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Rosenblatt, F. (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Roy, P. P., U. Pal, J. Lladós, M. Delalandre (2009). "Multi-Oriented and Multi-Sized Touching Character Segmentation Using Dynamic Programming." en. In: *2009 10th International Conference on Document Analysis and Recognition*. Barcelona, Spain: IEEE, pp. 11–15.

Saha, M. (2019). "GAN Based Text Line Segmentation." MA thesis. TU Kaiserslautern, Germany.

Sahu, D. K., C. V. Jawahar (Aug. 2015). "Unsupervised feature learning for optical character recognition." In: *13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1041–1045.

Schmidhuber, J., F. Gers, D. Eck (Sept. 2002). "Learning Nonregular Languages: A Comparison of Simple Recurrent Networks and LSTM." en. In: *Neural Computation* 14.9, pp. 2039–2041.

Schuster, M., K. Paliwal (Nov. 1997). "Bidirectional recurrent neural networks." en. In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis (Jan. 2016). "Mastering the game of Go with deep neural networks and tree search." en. In: *Nature* 529.7587, pp. 484–489.

Simistira, F., A. Ul-Hassan, V. Papavassiliou, B. Gatos, V. Katsouros, M. Liwicki (2015). "Recognition of historical Greek polytonic scripts using LSTM

networks." In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 766–770.

Sinha, A., M. Jenckel, S. S. Bukhari, A. Dengel (2019). "Unsupervised OCR Model Evaluation Using GAN." In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 1256–1261.

Smith, R. (2007). "An overview of the Tesseract OCR engine." In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. IEEE, pp. 629–633.

Springmann, U., D. Najock, H. Morgenroth, H. Schmid, A. Gotscharek, F. Fink (2014). "OCR of historical printings of Latin texts: problems, prospects, progress." In: *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pp. 71–75.

Srivastava, N., E. Mansimov, R. Salakhudinov (2015). "Unsupervised learning of video representations using lstms." In: *International conference on machine learning*, pp. 843–852.

Steels, L. (2008). "The Symbol Grounding Problemhas been solved. So what's next?" en. In: *Symbols and embodiment: Debates on meaning and cognition*, pp. 223–244.

Sutskever, I., R. Jozefowicz, K. Gregor, D. Rezende, T. Lillicrap, O. Vinyals (2015). "Towards principled unsupervised learning." In: *arXiv preprint arXiv:1511.06440*.

Sutskever, I., O. Vinyals, Q. V. Le (2014). "Sequence to sequence learning with neural networks." In: *Advances in neural information processing systems*, pp. 3104–3112.

Taddeo, M., L. Floridi (Dec. 2005). "Solving the symbol grounding problem: a critical review of fifteen years of research." en. In: *Journal of Experimental & Theoretical Artificial Intelligence* 17.4, pp. 419–445.

Tensmeyer, C., C. Wigington (2019). "Training Full-Page Handwritten Text Recognition Models without Annotated Line Breaks." In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 1–8.

Tolosana, R., R. Vera-Rodriguez, J. Fierrez, J. Ortega-Garcia (2018). "Exploring Recurrent Neural Networks for On-Line Handwritten Signature Biometrics." In: *IEEE Access* 6, pp. 5128–5138.

Vajda, S., Y. Rangoni, H. Cecotti (2015). "Semi-automatic ground truth generation using unsupervised clustering and limited manual labeling: Application to handwritten character recognition." In: *Pattern recognition letters* 58, pp. 23–28.

Williams, R. J., D. Zipser (1989). "A learning algorithm for continually running fully recurrent neural networks." In: *Neural computation* 1.2, pp. 270–280.

Williams, R. J., D. Zipser (1995). “Gradient-based learning algorithms for recurrent.” In: *Backpropagation: Theory, architectures, and applications* 433.

Wojna, Z., A. Gorban, D.-S. Lee, K. Murphy, Q. Yu, Y. Li, J. Ibarz (2017). “Attention-based extraction of structured information from street view imagery.” In: *arXiv preprint arXiv:1704.03549*.

Yousefi, M. R., M. R. Soheili, T. M. Breuel, D. Stricker (2015). “A Comparison of 1D and 2D LSTM Architectures for Recognition of Handwritten Arabic.” In: *DRR-XXI*. USA.

Yousefi, M. R., M. R. Soheili, T. M. Breuel, E. Kabir, D. Stricker (2015). “Binarization-free OCR for historical documents using LSTM networks.” In: *2015 13th international conference on document analysis and recognition (ICDAR)*. IEEE, pp. 1121–1125.

Yu, L., W. Zhang, J. Wang, Y. Yu (Aug. 2017). “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.” en. In: *arXiv:1609.05473 [cs]*. arXiv: 1609.05473.

Zhang, R., P. Isola, A. A. Efros, E. Shechtman, O. Wang (2018). “The unreasonable effectiveness of deep features as a perceptual metric.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595.

Zou, Y., R. V. Donner, N. Marwan, J. F. Donges, J. Kurths (Jan. 2019). “Complex network approaches to nonlinear time series analysis.” en. In: *Physics Reports*. Complex network approaches to nonlinear time series analysis 787, pp. 1–97.

Martin Jenckel

Resumé

Education

1999–2008 **Abitur**, *Robert-Stock-Gymnasium*, Hagenow, *1.4*.

2008–2011 **Bachelor of Science in Physics**, *Georg-August-University*, Göttingen, *2.1*.

2011–2012 **JYPE**, *Tohoku University*, Sendai.
Exchange Program

2012–2014 **Master of Science in Physics**, *Georg-August-University*, Göttingen, *1.4*.

2015– **PhD in Computer Science**, *TU Kaiserslautern*, Kaiserslautern.

Bachelor thesis

title *Heterogenous winding numbers in globally permutation symmetrically pulscoupled oszillators*

supervisors Prof. Dr. Marc Timme, apl. Prof. Dr. Ulrich Parlitz

Master thesis

title *Parameter estimation and critical point detection based on observed time series*

supervisors apl. Prof. Dr. Ulrich Parlitz, Honorarprofessor Dr. Stefan Luther

Work Experience

2014–2015 **Assitant Researcher**, *Max-Plank-Institute for Dynamic and Self-Organization*, Göttingen.

2015–2020 **Assitant Researcher**, *German Research Center for Artificial Intelligence*, Kaiserslautern.

Languages

Deutsch native

English conversational

Japanese rudimentary *A2-B1 level*