Document Image Processing for Handwritten Text Recognition

Deep Learning-based Transliteration of Astrid Lindgren’s Stenographic Manuscripts

RAPHAELA HEIL
Dissertation presented at Uppsala University to be publicly examined in Room 101121, Ångströmlaboratoriet, Lägerhyddsvägen 1, Uppsala, Wednesday, 4 October 2023 at 09:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Andreas Fischer (University of Applied Sciences and Arts Western Switzerland (HES-SO)).

**Abstract**


Document image processing and handwritten text recognition have been applied to a variety of materials, scripts, and languages, both modern and historic. They are crucial building blocks in the on-going digitisation efforts of archives, where they aid in preserving archival materials and foster knowledge sharing. The latter is especially facilitated by making document contents available to interested readers who may have little to no practice in, for example, reading a specific script type, and might therefore face challenges in accessing the material.

The first part of this dissertation focuses on reducing editorial artefacts, specifically in the form of struck-through words, in manuscripts. The main goal of this process is to identify struck-through words and remove as much of the strikethrough artefacts as possible in order to regain access to the original word. This step can serve both as preprocessing, to aid human annotators and readers, as well as in computerised pipelines, such as handwritten text recognition. Two deep learning-based approaches, exploring paired and unpaired data settings, are examined and compared. Furthermore, an approach for generating synthetic strikethrough data, for example, for training and testing purposes, and three novel datasets are presented.

The second part of this dissertation is centred around applying handwritten text recognition to the stenographic manuscripts of Swedish children’s book author Astrid Lindgren (1907 - 2002). Manually transliterating stenography, also known as shorthand, requires special domain knowledge of the script itself. Therefore, the main focus of this part is to reduce the required manual work, aiming to increase the accessibility of the material. In this regard, a baseline for handwritten text recognition of Swedish stenography is established. Two approaches for improving upon this baseline are examined. Firstly, a variety of data augmentation techniques, commonly-used in handwritten text recognition, are studied. Secondly, different target sequence encoding methods, which aim to approximate diplomatic transcriptions, are investigated. The latter, in combination with a pre-training approach, significantly improves the recognition performance. In addition to the two presented studies, the novel LION dataset is published, consisting of excerpts from Astrid Lindgren's stenographic manuscripts.

**Keywords:** document image processing, handwritten text recognition, stenography, strikethrough

Raphaela Heil, Department of Information Technology, Computerized Image Analysis and Human-Computer Interaction, Box 337, Uppsala University, SE-75105 Uppsala, Sweden.

© Raphaela Heil 2023

ISSN 1651-6214
URN urn:nbn:se:uu:diva-509138 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-509138)
To my family and friends
List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.


Reprints were made with permission from the publishers.
Summary of Contributions

The Roman numerals correspond to the numbers in the list of papers.

I Sole contributor to the paper idea, theoretical design, experiment planning and implementation. Significant contributions to the writing and conclusions.

II Sole contributor to the paper idea, theoretical design, experiment planning and implementation. Significant contributions to the writing and conclusions.

III Sole contributor to the paper idea and implementation. Significant contributions to the experiment planning, theoretical design, writing and conclusions.

IV Sole contributor to the paper idea, theoretical design, experiment planning and implementation. Significant contributions to the writing and conclusions.
Data annotations were kindly provided by volunteer stenographers. Preparation of experiment data by the author.
Related Work

In addition to the papers included in this thesis, the author has also contributed to the following works:

Associated (Peer-reviewed) Acts


Technical Reports


Datasets

https://doi.org/10.5281/zenodo.4767095

https://doi.org/10.5281/zenodo.4765063

https://doi.org/10.5281/zenodo.6406539

https://doi.org/10.5281/zenodo.8249845

Code

https://doi.org/10.5281/zenodo.4767169

https://doi.org/10.5281/zenodo.4767063

https://doi.org/10.5281/zenodo.8172293

https://doi.org/10.5281/zenodo.6406285

https://doi.org/10.5281/zenodo.7905299

# Contents

1 Introduction ............................................................................................................. 11
   1.1 Research Context ......................................................................................... 11
   1.2 Thesis Outline ........................................................................................... 11

2 Background ........................................................................................................... 13
   2.1 General Terminology .................................................................................. 13
      2.1.1 Characteristics of a Word ................................................................. 13
      2.1.2 Stenography vs Steganography ..................................................... 13
      2.1.3 Transliteration vs Transcription ............................................... 14
      2.1.4 Diplomatic Transcription ................................................................ 14
   2.2 Melin’s Stenography System ................................................................. 15
   2.3 Digital Image Processing ........................................................................... 17
      2.3.1 Digital Images .................................................................................. 17
      2.3.2 Binarisation .................................................................................... 18
      2.3.3 Morphological Operations .......................................................... 21
   2.4 Machine Learning Concepts ..................................................................... 21
      2.4.1 Types of Machine Learning ......................................................... 21
      2.4.2 Machine Learning Tasks ............................................................. 22
      2.4.3 Data Handling ................................................................................ 23
      2.4.4 Data Augmentation ....................................................................... 23
   2.5 Deep Learning ............................................................................................. 24
      2.5.1 Common Layers ............................................................................... 24
      2.5.2 Activation Functions ....................................................................... 25
      2.5.3 Loss Functions ................................................................................ 26
      2.5.4 Selected Architectures ................................................................... 27
   2.6 Metrics .......................................................................................................... 28
      2.6.1 Precision, Recall and F1 Score ...................................................... 29
      2.6.2 Root-Mean-Square Error ............................................................... 30
      2.6.3 Character and Word Error Rate .................................................... 30

3 Strikethrough Processing (Paper I and II) ......................................................... 32
   3.1 Obtaining Strikethrough Data .................................................................... 33
   3.2 Synthetic Strikethrough Generation .......................................................... 35
      3.2.1 Requirements and Assumptions ..................................................... 35
      3.2.2 General Approach ........................................................................... 39
      3.2.3 Perspectives .................................................................................... 42
   3.3 Datasets ...................................................................................................... 42
      3.3.1 IAM synth ..................................................................................... 42
3.3.2 Dracula_{\text{real}} .............................................................. 43
3.3.3 Dracula_{\text{synth}} ............................................................. 44
3.4 Strikethrough Identification and Classification ......................... 45
3.5 Unpaired Image to Image Translation (Paper I) ......................... 46
3.6 Paired Image to Image Translation (Paper II) ............................ 48
3.7 Conclusions and Perspectives .................................................... 51

4 Handwritten Stenography Recognition (Paper III and IV) ............... 54
4.1 The LION Dataset ....................................................................... 54
4.1.1 Data Splitting ........................................................................ 55
4.1.2 Editorial Marks ....................................................................... 56
4.2 Impact of Augmentations (Paper III) ........................................... 59
4.3 Integrating Stenographic Theory (Paper IV) ................................. 59
4.3.1 Considered Encoding Schemes ............................................. 61
4.3.2 Initial Encoding Results ......................................................... 62
4.3.3 Expanded Dataset Results ..................................................... 63
4.3.4 Impact of Editorial Marks ....................................................... 64
4.4 Conclusions and Perspectives ....................................................... 64

5 Conclusions and Future Directions .................................................. 67
5.1 Summary of Contributions .......................................................... 67
5.2 Future Directions .......................................................................... 68

Sammanfattning på Svenska ........................................................................... 69

Acknowledgements ....................................................................................... 71

References ........................................................................................................ 74

Skrifter utgivna av Svenska Barnboksinstitutet .............................................. 81

Appendix A: Exploring Reimplementation and Reproducibility of the
Optical Gregg Shorthand Recognition .......................................................... 85
In Brief: Original Method ........................................................................ 85
Scope of the Replication ........................................................................... 85
Reimplementation of Evaluation Metrics ................................................ 86
Reimplementation and Partial Evaluation of the Proposed Architecture .......... 86
Preliminary Conclusions and Future Work .............................................. 87
1. Introduction

1.1 Research Context

The research in this dissertation is largely influenced by the project *The Astrid Lindgren Code*\(^1\), which is centred around the original manuscripts of Swedish children’s book author Astrid Lindgren (1907 – 2002), written in stenography, or shorthand. One of the aims of *The Astrid Lindgren Code* project is to make the manuscript contents available, using approaches from handwritten text recognition (HTR), and employing expert crowdsourcing [4], primarily in the form of trained stenographers. In addition to this, literary research questions, with particular focus on revisions within the drafts, are investigated, based on the obtained transliterations.

This project offers several interesting opportunities from a document image processing (DIP) and HTR standpoint. One of these is the application of DIP and HTR to stenography, an area of research that was established in the 1980s (cf. e.g. the work by Leedham and Downton [49]), but has attracted limited attention over the past 40 years. In addition to this, the project’s interest in manuscript revisions poses the challenge of automatically processing indicators of editorial activities, such as strikethrough, and additions or alterations. This task of recognising altered texts, despite the presence of potential obfuscations, has, to the best of my knowledge, not been a concern in other HTR-related projects.

The contributions of this dissertation, in the context of *The Astrid Lindgren Code* project, are twofold: Firstly, in Papers I and II, approaches for accessing texts, altered by strikethrough, are examined. Secondly, in Papers III and IV, the data obtained via crowdsourcing is leveraged to establish an HTR baseline for stenography and to investigate different techniques for data augmentation and encoding of transliterations. A third, more general, contribution is the publication of related datasets and code for each of the four papers, with the goal of promoting open science, open data, and reproducibility.

1.2 Thesis Outline

This dissertation consists of five main chapters. Following this introduction, the technical background is introduced in chapter 2. Chapter 3 presents the

\(^1\)Funded by Riksbankens Jubileumsfond (RJ), reference number: P19-0103:1
work that has been performed regarding the handling of strikethrough, covering Papers I and II, while chapter 4 discusses text recognition in the context of stenography, i.e. Papers III and IV. Lastly, conclusions and perspectives of this dissertation are discussed in chapter 5.
2. Background

This chapter introduces foundational concepts that are of relevance to this dissertation. Following a presentation of general terminology (section 2.1) and Melin’s stenographic system (section 2.2), digital image processing (section 2.3), machine learning (section 2.4) and deep learning (section 2.5) concepts will be introduced. The chapter concludes with a brief presentation of relevant metrics (section 2.6).

2.1 General Terminology

In the following, selected terms are introduced and delineated.

2.1.1 Characteristics of a Word

The visual representation of a word, be it handwritten or printed, can be described by a number of features and terms. Selected ones are introduced in the following, and visualised in Figure 2.1.

Considering the vertical extent, a word can be divided into up to three regions. Firstly, the core, that forms the centre of the word. Secondly, ascenders, i.e. parts of characters that extend upwards from the upper edge of the core region, and thirdly descenders, extending below the core. Ascenders and descenders may not always be present, considering for example the word “moon”, which contains neither. The lower edge of the core region is generally referred to as the baseline of a word.

Lastly, the stroke width describes the general stroke thickness of a written word. In the case of handwritten words, the stroke width may vary across the trajectory of a given word and depends on the writing implement, and, to some degree, the individual style of the writer.

2.1.2 Stenography vs Steganography

A literature survey based on the term *stenography* yielded a considerable number of works related to *steganography*. In order to avoid any confusion, both terms are briefly introduced below.

*Stenography*, also referred to as shorthand, describes the practice of using a specifically designed script, consisting of short strokes, to collect handwritten notes at high speeds. Historically, shorthand has been used in places like
Figure 2.1. Visualisation of selected terms, describing aspects of a written word.

courthouses and offices, for example by secretaries, to quickly record relevant pieces of information [4].

Steganography describes the process of hiding information within physical, or digital, objects [13]. A simple example, that the reader may be familiar with from their childhood, is the writing with “invisible ink”, such as lemon juice, which becomes visible when exposed to reasonable heat.

2.1.3 Transliteration vs Transcription

Transliteration defines the process of representing letters from one alphabet or language in another [1]. This may also entail that a single letter is transliterated as more than one, for example in the case of the German “ß” that can be represented as a double “s” for fonts that do not support the former symbol.

In the context of handwritten text recognition, the term transcription is generally used to describe representations within the same alphabet or language, for example when digitising a Latin manuscript with Latin letters (cf. e.g. [24, 73]).

In this thesis, the term transliteration is primarily used, as stenography presents its own alphabet, with special letter forms, which are transliterated into Swedish characters.

2.1.4 Diplomatic Transcription

Diplomatic transcriptions expand upon the aforementioned transcription process by extending the target symbol set with additional letters, beyond the
language’s standard alphabet. These additional forms are used to minutely record different symbols, that are being used in the source text. For historic documents, this for example entails differentiating between the regular, also called *round*, “s” and its long counterpart\(^1\), shown in Figure 2.2. Another example, which is still in use today, is the *ampersand*, \&, denoting an *and*. In the former example, the diplomatic transcription would use two different symbols, one per shape of s. For the latter case, instead of expanding the abbreviation and transliterating the ampersand as *and*, the \&, or another previously unassigned symbol, may be used in the diplomatic transcription.\(^{[66]}\)

2.2 Melin’s Stenography System

Chapter 4 deals extensively with the recognition of Astrid Lindgren’s manuscripts, written in the Swedish stenography system that was proposed by Olof Melin (1861–1940) in the 1880s. Like other stenography systems, it aims to provide an efficient and fast way for writing, and is based on the combination of short strokes, hooks and loops, that represent different letters and letter combinations\(^{[26]}\). Figure 2.3 shows the words “jonatan” and “nangijala”\(^2\) written in Melin’s system. The strokes are coloured to roughly indicate the extent and shape of each of the symbols. Note that Melin’s system does not differentiate between upper and lowercase letters and that it is generally written in a connected form.

Since several characteristics of stenography will be considered in this work, a brief overview of the writing system is provided below. The following sections are not intended, or designed, as learning material and the interested reader is referred to the literature, for example\(^{[2]}\), and the Swedish society for Melin’s stenography\(^3\).

---

\(^1\)Used for example instead of the first round “s”, in cases where a double-consonant appears, like in “congre[s]”.

\(^2\)One of the main characters, and places, respectively, from Lindgren’s novel *The Brothers Lionheart*.

Figure 2.3. Two examples for words, written in Melin’s stenography system. Colours indicate stroke segments, corresponding to word segments in the transliterations below. Vowels are indicated by the same colour to emphasise the high degree of visual similarity.

Figure 2.4. Melin symbols and their corresponding letters in the Swedish alphabet.

Regular Alphabet

Melin’s system firstly defines one symbol per character in the Swedish alphabet, shown in Figure 2.4. It can be noted that several of the symbols are written in a similar fashion, which is especially noticeable for characters that are also phonetically similar, such as “e” and “ä”, and “d” and “t”.
Compound Symbols

Character compounds, or n-grams, that are frequently used in the Swedish language, have been assigned their own symbols. An example for this is the “ng” in “nangijala” in Figure 2.3. Furthermore, a selection of compound symbols is shown in Figure 2.5.

\[ sj \quad tj \quad ns \quad nj \quad sn \]

Figure 2.5. Selected compound symbols – the complete list differentiates between more than 20 different combinations and corresponding signs.

Shortforms

In addition to the two previous groups of symbols, Melin’s system uses shortforms, i.e. frequently-used words that are represented by short strokes – hence the name. Shortforms exist in a number of variations, with a large portion representing individual words. Others are used to indicate portions of words, such as suffixes. Furthermore, some of the character symbols, shown in Figure 2.4 are used both, to represent the shown character, as well as an individual word or subword. Figure 2.6 shows examples for the different types of shortforms, including the aforementioned “character shortforms”, “j[ag]” and “o[ch].

\[ aldeles \quad -are \quad j[ag] \quad o[ch] \]

Figure 2.6. Selected shortforms; the two symbols on the right are used both for the characters “j”, respectively “o”, as well as for the words “jag” and “och”.

2.3 Digital Image Processing

This section introduces a number of classical digital image processing concepts, that are relevant in this dissertation.

2.3.1 Digital Images

In the context of document image processing, digital images are generally acquired using cameras, such as digital single-lens reflex cameras (DSLRs), or scanners\(^4\). The former are typically mounted in a top-down copy stand, as

\(^4\)Although rather rare, volumetric scans, for example via micro–computed tomography, may also occur[77].
shown in Figure 2.7, to ensure stable images. Scanners are used in different variations, ranging from commonly-known office equipment (e.g. flatbed scanners) to more specialised varieties, such as book scanners. Depending on the manuscript’s material and content, and the imaging setup, the digital images may cover single pages, portions thereof, or book spreads, i.e. double pages. Furthermore, the images may contain other aspects of the material, such as bindings and page markers, as well as portions of the imaging background at the edges. Figure 2.8 shows an example page from Astrid Lindgren’s stenographic manuscripts\(^5\), containing binding at the top, and the aforementioned imaging background, towards the edges of the image. Artefacts, such as the ones shown here, are often removed as part of the image processing pipeline, as these are not relevant to downstream tasks, such as text recognition.

Digitally, the acquired image is represented as a regular grid of data. Each cell in the grid, referred to as pixel – short for picture element – contains a number of values, arranged in channels, presenting a discretised version of the image information. This dissertation is only concerned with three- and single-channel images, although other arrangements, also with respect to the grid structure, exist in other areas of digital image processing. The most common form of three-channel images is referred to as RGB – denoting the three channels red, green and blue – which encode the intensities of the respective colour. Single-channel images may either consist of two discrete values, representing a binary image (typically black and white), or of a range of values, corresponding to the greyscale intensity. Figure 2.9 demonstrates different combinations of colours and intensities (top), the separation of the three channels R, G and B, as well as possible greyscale and binary versions.

### 2.3.2 Binarisation

Document images can display large variations with respect to the colour and texture of the background and text. A commonly used approach to reduce this variation is to binarise the images, i.e. to group the pixels into two classes, typically the background and text. A popular classical approach in this category is Otsu’s method [62], which defines a global threshold, i.e. intensity level, at which to separate the greyscale pixels into foreground and background. An example for this is shown in Figure 2.10.

The task of binarising a diverse range of document images has been investigated for a long time in the DIP community [82], and challenges in this area are frequently being hosted at conferences [70]. While binarisation is still a relevant preprocessing step for classical approaches (cf. e.g. [28]), contemporary deep learning methods often rely on greyscale images instead (cf. e.g. [45, 73]).

\(^5\)acquired with a DSLR and copy-stand
Figure 2.7. Example of a top-down copy stand setup, used here to digitise a photo slide. Image credit: flickr-user pedrik (Pedro Mendes), CC-BY 2.0 (2018); https://flickr.com/photos/24388834@N04/39575554842

Figure 2.8. Digitisation example from the Astrid Lindgren corpus, showing the binding at the top and the imaging background towards the edges.
Figure 2.9. Top: colour image, triplets indicate the area’s intensity with respect to each of the three channels – red, green and blue – (range: \([0, 1]\)); middle: top image, separated into the three colour channels; bottom: greyscale version (left) and Otsu-binarised [62] (right) versions of the top image.

Figure 2.10. Left: input image, right: binarisation result, using the threshold obtained from Otsu’s method [62]. Note the removal of texture for the dark background and light stroke.

Figure 2.11. Original image (left) and its greyscale dilation (middle) and erosion (right), using a disk structuring element with radius 2 px.
2.3.3 Morphological Operations

Morphological operations are a group of image processing methods that can for example be applied as a preprocessing step to enhance (or reduce) certain features. In this dissertation, the two operations dilation and erosion are applied to greyscale images in the context of data augmentation (cf. subsection 2.4.4). Greyscale dilations result in a thickening of bright (foreground) features, as shown in Figure 2.11 (middle). This operation is defined as:

\[
[f \oplus b](x, y) = \max_{(s,t) \in \hat{b}} \{f(x-s,y-t)\}
\]  

(2.1)

where \( f \) is the image that is being dilated by the flat structuring element \( b \), and \( \hat{b} \) is the window, spanned by \( b \).

In contrast to this, as demonstrated on the right in Figure 2.11, greyscale erosions result in a thinning of bright structures. This operation is defined as:

\[
[f \ominus b](x, y) = \min_{(s,t) \in b} \{f(x+s,y+t)\}
\]  

(2.2)

again, with the image \( f \) and flat structuring element \( b \). [81]

2.4 Machine Learning Concepts

The following sections introduce selected machine learning concepts, covering types of machine learning (subsection 2.4.1), tasks (subsection 2.4.2), data handling (subsection 2.4.3) and finally data augmentation (subsection 2.4.4).

2.4.1 Types of Machine Learning

Machine learning approaches can be grouped, based on the configuration of data that is being used to train them. Broadly, they can be divided into the following three categories.

**Supervised Learning**

In a supervised data setting, each datapoint has a corresponding target representation, in line with the task at hand. This can for example be a label, a target transcription or a corresponding target image. The target information is available during training, to improve and monitor a model’s performance, as well as afterwards, to evaluate how well a given model performs on the task [29].

**Unsupervised Learning**

In unsupervised learning, as the name indicates, no information about a potential target representation is connected to the input datapoints [29]. In such
settings, a machine learning model may for example aim to establish a clustering, grouping datapoints that share similarities with respect to certain features (cf. e.g. [80]). Unsupervised data settings may for example occur in cases where data annotation is expensive or tedious.

**Semi-Supervised Learning**
As a middle-ground between the two former data settings, semi-supervised learning is based on data that is largely without annotations, except for a small portion. The limited amount of annotated data may for example entail a few samples per label, that serve as reference labels for other datapoints within the same cluster, such as in [7].

2.4.2 Machine Learning Tasks
Machine learning can be applied to solve a variety of tasks. Three of these, which are central to this dissertation, are introduced in the following.

**Classification**
In a classification setting, a given machine learning model aims to assign a class, out of a group of class options, to a presented datapoint [54]. Such a classification can be binary, i.e. consisting of two possible classes, for example indicating whether a given image shows handwritten or printed text. However, much larger numbers of classes are also conceivable, as is, for example, the case in the *ImageNet Large Scale Visual Recognition Challenge* [22, 75], which spans 1000 classes.

**Image-to-Image Translation**
Image-to-image translation approaches are concerned with transforming an image from a given source domain to a certain target domain [29]. Popular examples for this are the rendering of a photography in the style of a famous painter [27], such as Van Gogh, and the transformation of horses to zebras [91]. This group of methods can also be used in the context of denoising, for example, to remove motion blur from transmission electron microscopy images [85].

Image-to-Image translation can either be performed in a paired or unpaired data setting. In the former case, the model is trained, using aligned pairs of data, for example a landscape image taken from the same location by day and at night [42]. The unpaired data arrangement generally consists of a high-level grouping of images into the source and target domains but without a one-to-one correspondence between individual samples. To illustrate this, the reader may consider the earlier example of horse and zebra images: while it is certainly possible to collect images of either, perhaps even in similar poses, it will never be possible to collect real-world images with a pixel-perfect correspondence.
In this task setting, data, such as images or an audio signal, is transferred into a textual representation, consisting of a sequence of symbols [29]. A considerable portion of DIP falls into this particular category, with topics such as Optical Character Recognition (OCR, e.g. [23]) and Handwritten Text Recognition (HTR, e.g. [5, 21]), both cases in which text is extracted from images.

2.4.3 Data Handling

The data that is used to train and evaluate a machine learning model, is typically divided into at least three separate, and non-overlapping, sets, termed train, validation, and test. As the name indicates, the train portion is used to train a model, i.e. update its learnable parameters. The validation and test set are both used to evaluate a trained model, but at different times in the learning process and for different purposes. A validation set is generally used to fine-tune various hyperparameters, such as a model’s size or training duration. This set is therefore part of the training process and its choice and composition may thus have an impact on the model’s final performance, as it determines which instance of a model to retain. In order to obtain a measure of a model’s performance on unseen data, the held-out test set is used for the final evaluation.

In cases where the entirety of available data is small, cross validation may be applied to obtain an estimate of a model’s performance, respectively to choose suitable hyperparameters. Under a k-fold cross validation setting, the data is split into \( k \) separate partitions, or folds. In turn, each of the folds is held back, as unseen evaluation set, while the \( k - 1 \) other folds are used to train the model [29].

2.4.4 Data Augmentation

Data augmentation is commonly, but not exclusively, used in low-resource data settings. It aims to increase the amount, and potentially diversity, of the training data, by expanding this set with slightly altered versions of the original datapoints [78]. In an imaging context, this often entails the application of transformations, such as random rotations and scaling (e.g. [89]). Other examples of image augmentations are morphological operations (cf. subsection 2.3.3) [86], and different types of noise and colour transformations [78].

Besides these augmentations, that create datapoints by modifying existing ones, some methods have been proposed that generate entirely new data within a given domain, for example by employing Generative Adversarial Networks (GANs) [3, 44, 56].
2.5 Deep Learning

Deep learning is a form of machine learning that is centred around models that consist of deep stacks of layers with learnable parameters. The following sections introduce selected components of deep learning models, followed by a presentation of selected architectures, that are employed in this dissertation.

The literature on this very active field of research is vast and the following sections therefore only present a snapshot of the topics that are most relevant for this dissertation. For an in-depth introduction to deep learning, see for example [29].

2.5.1 Common Layers

Layers, containing trainable parameters, are the main building blocks of a deep neural network. A few selected ones are introduced below.

**Fully Connected Layer**

Fully connected layers, also referred to as linear layers, apply a linear transformation to the input, i.e.

$$y = Wx$$  \hspace{1cm} (2.3)

with the input $x \in \mathbb{R}^m$, the output $y \in \mathbb{R}^n$ and the learnable parameters $W \in \mathbb{R}^{n \times m}$.

In modern deep learning, fully connected layers, for example, find applications as the final layer in convolutional neural networks (CNNs) for classification, where the output size is equal to the number of possible classes (cf. e.g. [34]).

**Convolutional Layer**

Convolutional layers consist of convolving an image with a set of learnable filters, i.e.:

$$y = x \ast g$$  \hspace{1cm} (2.4)

with input $x \in \mathbb{R}^{m \times n \times c}$, where $m$ and $n$ are the input (e.g. image) height and width and $c$ is the number of channels, convolutional filter $g \in \mathbb{R}^{a \times b \times d}$, and output $y \in \mathbb{R}^{e \times f \times d}$ [47]. In the basic form, the shape of the convolutional output, i.e. $e$ and $f$, are calculated via: $e = m - a + 1$, respectively $f = n - b + 1$. Many implementations offer options that may further affect the output size, for example padding, i.e. expanding the input around the edges, and striding, i.e. moving the convolutional filter with a step-size greater than one [67].

**Recurrent Layers**

Recurrent layers are used to process and produce sequential data, each often of variable length. In this general type of layer, information from the previous
timestep may be passed into the calculation for the next position in the output sequence, via recurrent connections. The two concrete recurrent layers that are relevant to this dissertation are referred to as Long Short-Term Memory (LSTM) [37] and Gated Recurrent Units (GRUs) [20]. In both cases, the layers are constructed from an elaborate set of gates, i.e. functions that determine which, and how much, of the input and previous information are employed in the calculation of the next output.

2.5.2 Activation Functions
Activation functions are interspersed in-between the aforementioned layers, and introduce non-linearity into the network. The choice of the final activation function, i.e. producing the model’s output, is dictated by the model’s task. A few popular activation functions are introduced below.

Rectified Linear Units
Rectified Linear Units (ReLUs) [25, 59] are a popular choice of activation function and are primarily used in combination with the inner (hidden) layers of a model. This activation is defined as:

\[ f(x) = \max(0, x) \] (2.5)

Parametric Rectified Linear Units
Parametric Rectified Linear Units (PReLUs) [33] extend the concept of ReLUs by scaling negative inputs by a learnable parameter \( a \), instead of defaulting to zero:

\[ f(x) = \max(0, x) + a \cdot \min(0, x) \] (2.6)

where \( a \) can be defined across the entire output or differ for portions of it, for example channel-wise.

Sigmoid
The sigmoid activation is defined as:

\[ f(x) = \frac{1}{1 + e^{-x}} \] (2.7)

This activation produces outputs in the range \([0, 1]\), which can for example be interpreted as pixel intensities in image-to-image translation settings.

Softmax
The softmax activation is frequently used as final layer in classification settings, where the output is interpreted as class probabilities. It is defined as:

\[ f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \] (2.8)
2.5.3 Loss Functions

Loss, or objective, functions, are used to calculate the error between a model’s prediction $\hat{y}$ and the target $y$, i.e. the expected output. In deep learning, the goal is typically to minimise the loss by updating the model’s learnable parameters, for example via gradient descent \[29\]. Below, the primary losses that are used in Papers I - IV, are briefly introduced.

$L_1$-Loss

The $L_1$-Loss, also referred to as the mean absolute error (MAE), is defined as:

$$
L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|
$$

(2.9)

It is used in Paper I as part of the cycle consistency (cf. section 2.5.4).

$L_2$-Loss

The $L_2$-Loss, also referred to as the mean squared error (MSE), is defined as:

$$
L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2
$$

(2.10)

This loss is used in Paper I for the discriminator training (cf. section 2.5.4).

Cross-Entropy

Cross entropy provides a measure for the difference between two probability distributions and is frequently used in classification tasks. For a classification task with $n$ classes, it is defined as:

$$
L(y, \hat{y}) = -\sum_{i=1}^{n} y_i log(\hat{y}_i)
$$

(2.11)

In the pixel-wise, binary form, it is for example used in autoencoder settings (cf. section 2.5.4):

$$
L(y, \hat{y}) = y log(\hat{y}) + (1 - y) log(1 - \hat{y})
$$

(2.12)

In the context of this dissertation, cross entropy is used for the classification tasks in Paper I, while binary cross entropy is used for the autoencoder-based reconstruction in Paper II.

Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) was introduced in 2006 by Graves et al. [31] and has since found frequent applications in HTR (cf. e.g. [73]). It maps a predicted sequence to a target representation, where the former is expected to be longer or of same length as the latter. During the mapping,
a collapsing function \( \mathcal{B} \) is applied that removes all character repetitions, unless they are separated by a special CTC blank symbol. For a single pair of predicted and target sequences, the loss is defined as:

\[
\mathcal{L}(y, \hat{y}) = -\log \sum_{A \in \mathcal{B}^{-1}(y)} \prod_{t=1}^{T} p(a_t | \hat{y})
\]  

(2.13)

where \( \mathcal{B}^{-1} \) is the inverse of the collapsing function, producing all sequences \( A \) that would result in the target representation, when collapsed.

Section 2.5.4 provides further details on how the CTC can be used in an HTR setting and how the final transcription is obtained from such a model. This setup is used in Papers III and IV.

2.5.4 Selected Architectures

There is a large variety of deep learning architectures, many of which are specifically designed for a certain use case or task domain. Below, we introduce three general types of deep neural networks, that are used in this thesis. Firstly, Autoencoders, which are part of Papers I and II, secondly Cycle-Consistent Generative Adversarial Networks, for Paper I, and lastly, Convolutional Recurrent Neural Networks, for Papers III and IV.

Autoencoders

Autoencoders are a group of neural networks that are trained to reproduce a high-dimensional input from a lower-dimensional, intermediate representation [11]. The typical structure of an autoencoder consists of an encoder that performs the dimensionality reduction, for example via convolutions, and an often symmetric decoder, that operates in the opposite direction. The representation, obtained from the bottleneck between encoder and decoder, can for example be used in data compression scenarios [18].

Instead of reproducing the input, an autoencoder can also be trained to produce a different target representation. Example applications for this are document binarisation [14] and denoising [90].

Cycle Consistent Generative Adversarial Networks

Cycle-Consistent Generative Adversarial Networks (Cycle-GANs) were originally proposed for unpaired image-to-image translation [91] and consist of two sets of conditional generative adversarial networks (GANs) [30, 57]. Each conditional GAN in turn consists of a generator \( G \) and a discriminator \( D \). The generator, which often takes the shape of an autoencoder, is trained to translate the input image from the source to the target domain. Additionally, the corresponding discriminator receives both real and generated images from the target domain and is trained to classify whether a given input is fake or not.
These two components are trained in tandem, with each trying to outperform the other. As indicated earlier, a CycleGAN consists of two such GANs, with one translating images from domain A to B, and the other from B to A. In this setting, the regular GAN loss is furthermore extended with a cycle-consistency constraint, where an image, that is passed through both GANs, i.e. translated from domain A to B, and then back to A, should be similar to the original input.

**Convolutional Recurrent Neural Networks**

At the time of writing, convolutional recurrent neural networks (CRNNs) [10, 71] are a popular choice of architecture in HTR [73]. They generally consist of a convolutional backbone, followed by an LSTM [37] or GRU [20] head, and are trained using the CTC loss [31, 32]. In this setup, the CNN acts as a feature extractor, producing a feature vector per timestep, with each timestep essentially corresponding to a slice of columns of the input image. Based on these features, the recurrent head, potentially in combination with a fully-connected layer, produces an output with one row per character in the model’s alphabet and one column per timestep. An example for this kind of output, taken from one of the models from Paper IV, is shown in Figure 2.12. In order to obtain the final transcription from this representation, a decoding step has to be applied. Several different options, such as Word Beam Search [76] and language model-based approaches [21] exist for this.

The most straightforward decoding method, which does not require additional language information\(^6\) such as dictionaries, is best-path, also called greedy, decoding. For this approach, each timestep is transcribed as the symbol with the highest CTC score at that point in the model output. As can be seen in Figure 2.12, this may result in unwanted character repetitions. The obtained sequence is therefore further processed by applying a collapsing function that removes any superfluous, repeated characters. In order to still be able to obtain transliterations with double-characters, as for example in “Uppsala”, the aforementioned CTC blank character is added to the alphabet, resulting for example in “UUpppp#psaalaaa”, with # representing said blank. Collapsing all repeated characters yields “Up#psala”, which is converted into the final transcription, “Uppsala”, by removing all CTC blanks.

2.6 Metrics

Below, relevant metrics are introduced to provide an easily accessible reference while reading this dissertation.

---

\(^6\)Approaches that incorporate language information have the potential to considerably outperform best path decoding, for example by counteracting spelling mistakes with language knowledge. They do however come at the cost of requiring a suitable corpus for initialisation, which may not always be feasible.
Figure 2.12. Output of a CTC-based model. Output has been truncated for visualisation purposes after applying a softmax function. Missing values are indicated by “...”, original height corresponds to the alphabet size, here 51. Raw transcription reads: “jjo####nnnaattann####” (# indicates the blank), post-processed transcription: “jonatan”.

2.6.1 Precision, Recall and F\textsubscript{1} Score

For binary classification, with a positive and a negative class, precision (P), recall (R) and the F\textsubscript{1} score are frequently-used metrics. These are defined as follow:

\[
P = \frac{TP}{TP + FP} \quad \text{(2.14)}
\]

\[
R = \frac{TP}{TP + FN} \quad \text{(2.15)}
\]

\[
F_1 = \frac{2 \times P \times R}{P + R} \quad \text{(2.16)}
\]

where $TP$ refers to the number of true positives, $FP$ the false positives and $FN$ the false negatives.

These metrics can also be applied when, for example, comparing binary word images, as we do in Papers I and II, where the foreground colour – generally the ink – is considered as the positive, and the background as the negative class.

Besides this, $F_1$ scores can also be considered in multiclass settings by aggregating the results for individual classes. The aggregation method that is relevant in the context of this dissertation is the *macro* $F_1$ score, which is calculated as the unweighted mean of the per-class $F_1$ scores.
2.6.2 Root-Mean-Square Error

The root-mean-square error (RMSE) between a target image \( y \) and the predicted image \( \hat{y} \), with \( N \) pixels each, is defined as:

\[
RMSE(y, \hat{y}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2}
\]  

(2.17)

2.6.3 Character and Word Error Rate

The text recognition performance of a model is commonly assessed via the character and word error rates (CER, respectively WER). These are based on the Levenshtein distance \([50]\), i.e. the minimum number of insertions (I), deletions (D) and substitutions (S) that are required to transform a given string into the reference string. For the CER, the Levenshtein distance is calculated at a character, and for the WER at a word level. The obtained distance is normalised by the number (N) of characters, respectively words, in the reference string, i.e.:

\[
ER = \frac{S + D + I}{N}
\]  

(2.18)

**Example**

To demonstrate the calculation of CER and WER, consider the following ground truth sequence “Document Image Processing HTR” and model output “Document Processing HSR Lindgren”. As indicated in Figure 2.13 (top) six insertions (five characters, one space), nine deletions (eight characters, one blank), and one substitution are required at a character level, in order to transform the model’s output into the ground truth text. This results in a CER of \((6 + 9 + 1)/29 \approx 0.5517\). Figure 2.13 (bottom) indicates the corresponding transformations at a word level, requiring one substitution, insertion and deletion each, resulting in a WER of \(3/4 = 0.75\).
Character-level:
Document Image Processing HTR
Document Processing HSR Lindgren

Word-level:
Document Image Processing HTR
Document Processing HSR Lindgren

Insertion Substitution Deletion

*Figure 2.13.* Example, highlighting the operations performed at a character (top) and word-level (bottom) to transform the predicted sequence “Document Processing HSR Lindgren” into the target representation “Document Image Processing HTR”.
3. Strikethrough Processing (Paper I and II)

Figure 3.1. Example demonstrating the vast use of strikethrough in Astrid Lindgren’s manuscripts.

One of the characteristics of Astrid Lindgren’s manuscripts that sets them apart from many of the publicly available HTR datasets, is the large amount of struck-through words. Generally, strikethrough indicates a writer’s intention to delete a word or larger portion of text, for example to replace it with an altered version. It is therefore not surprising that a manuscript collection like the Astrid Lindgren archive, which contains drafts from various editorial stages, exhibits a large number of such corrections. What sets Lindgren’s strikethrough apart from that appearing in many other datasets, for example the IAM database [55], is that the struck-through content is of interest to literary scholars. An example for this is the field of genetic criticism, which studies a piece of literature by taking its drafts and revisions into account [41]. As literary research questions like these are part of the Astrid Lindgren Code project, struck-through words should not be discarded, but transcribed in the same fashion as the rest of the text.

In the context of document image processing, two main paths of approaching the correct recognition of struck-through words are generally conceivable:

a) preprocessing the struck-through portions to remove as much (ideally all) of the disturbance, leaving behind a more readable word

b) training an HTR model to directly, and correctly, recognise struck-through words, despite the alterations

In this chapter, and the related Papers I and II, we are following the first, preprocessing-based approach for two main reasons. Firstly, when the task
of recognising struck-through words became apparent in the Astrid Lindgren Code project, the crowdsourcing of ground-truth transliterations, that are required for b), had not yet progressed far enough to train any form of recognition model. Secondly, the preprocessing in a) is not only relevant for machine-based approaches, but can also aid human transliterators in deciphering struck-through words, for example as part of an annotation tool.

The following sections present an overview and discussion of the work performed in Papers I and II. Firstly, in section 3.1, we briefly outline the challenges related to collecting strikethrough data, and present the different approaches that have been taken so far. Afterwards, we introduce our approach for generating synthetic strikethrough images (section 3.2), followed by the presentation of relevant datasets (section 3.3). Sections 3.4 and 3.5 summarise the work performed in Paper I. The former is concerned with the identification of struck-through words and the classification of different stroke types, while the latter presents our initial approach to removing strikethrough, employing CycleGANs. Section 3.6 introduces our follow-up work, which investigates the use of paired image-to-image translation approaches and compares these to CycleGANs. Lastly, conclusions regarding strikethrough removal, and avenues for future work are presented in section 3.7.

3.1 Obtaining Strikethrough Data

Pairs of struck-through images and their clean counterparts are not strictly necessary for developing cleaning approaches, as we will for example discuss in section 3.5. They are however crucial for the precise evaluation and comparison of different methods. The main problem with collecting such paired datasets is that strikethrough strokes essentially destroy the clean image, rendering the true original unobtainable.

To the best of our knowledge, four general approaches have been proposed to date, in order to create paired strikethrough data. These are briefly introduced in the following:

*Aligning Clean and Struck Versions*

Firstly a collection of clean words is written out and scanned, followed by the application of strikethrough and a second scan of the same documents. The two obtained versions are aligned, and for example segmented, yielding the paired images [36]. While this approach will result in an aligned dataset, containing genuine strikethrough images, it requires a large amount of human labour, both for the writing and striking-through of words. Furthermore, it is not directly applicable when creating a paired dataset as basis for cleaning a specific corpus, which investigates the use of paired image-to-image translation approaches and compares these to CycleGANs. Lastly, conclusions regarding strikethrough removal, and avenues for future work are presented in section 3.7.

---

1Proxy measures, such as recognition performance, could be used instead (cf. [69]). While these provide other interesting insights, for example the impact of a cleaning approach on downstream tasks like text recognition, they can only provide an approximation of the cleaning performance.
for which an in-domain dataset may be preferred. The method can be adapted to such a use case by, for example, striking out words on a copy of the documents of interest. This may, however, introduce inconsistencies, for example due to printing and scanning artefacts or by using a different writing tool than the original scribe.

**Manual Cleaning**

Struck-through images are cleaned by removing the pixels that belong to the strikethrough stroke, for example as part of an interactive pipeline, as proposed by Chaudhuri and Adak [16]. This approach yields a dataset of genuine strikethrough-clean pairs and can be applied to any already existing corpus. The main drawback of this method is the considerable human involvement that is required, making it potentially expensive to implement. In addition to this, a limiting factor of this approach is that it can only produce as many image pairs as there are struck-through words in the original dataset. Considering for example the IAM database [55], only 55 out of 115 320 words are struck-through, thus yielding a very small set of relevant pairs.

**Superimposing Genuine Strokes**

Poddar et al. [68, 69] present an approach that employs genuine stroke images, collected separately from any handwritten text, which are then superimposed onto a given word. In contrast to the previous two approaches, this requires a minimal amount of human labour, as different strokes can be collected quickly on an empty piece of paper. A potential concern here is that the strokes are collected separately from the actual words, which could lead to clear visual differences that in turn might affect the generalisation to the actual target domain. This concern can likely be mitigated to a large degree by ensuring that the stroke collection setting is similar to that of the target document, for example by using a similar pen.

**Superimposing Synthetic Strokes**

The fourth approach is similar to the previous one, however instead of superimposing images of genuine, handwritten strokes, synthetic ones are used [36, 51]. These stroke images are automatically generated based on certain constraints, such as stroke type, size and colour. For this method, similar concerns as with the previous one apply, as the generation may lead to strokes that are very different from the ones naturally appearing in the target data. Depending on the concrete stroke generation algorithm, this may be mitigable by tuning or limiting certain parameters. A clear advantage of this approach is that it does not require human intervention and can generate large amounts of varying
strokes quickly.

As outlined above, the approaches present different requirements with respect to data availability and access to human annotators. In Papers I and II we focus on the first and last approach, i.e. aligning images collected before and after the application of strikethrough (cf. section 3.3.2), and generating synthetic strikethrough (cf. section 3.2). The latter choice was inspired by the work of Likforman-Sulem and Vinciarelli [51] and was primarily made due to limited access to human annotators\(^2\). In addition to this, the former approach was taken in order to supplement the synthetic evaluations and demonstrate the extent to which the strikethrough-removal, trained on synthetic data, translates to genuine strikethrough.

### 3.2 Synthetic Strikethrough Generation

This section presents the synthetic strikethrough generation approach that we developed as part of Paper I. We begin with a discussion of requirements and assumptions that our method is founded on (subsection 3.2.1), followed by a presentation of our general approach (subsection 3.2.2). Lastly, we outline opportunities for future work regarding synthetic strikethrough generation (subsection 3.2.3).

#### 3.2.1 Requirements and Assumptions

The development of our synthetic strikethrough generation approach was guided by a number of requirements and assumptions that are summarised below.

**General Requirements**

As a first general requirement, we determined a set of stroke types that should be generated by our approach. We base this list on the one presented by Chaudhuri and Adak [16] and extend it by an additional entry. Concretely, we consider the following seven types – phrases in parentheses indicate the name by which each stroke is referred to in the remainder of this work; examples are shown in Figure 3.2:

- a single horizontal line (*single*)
- two horizontal lines (*double*)
- a single diagonal line (*diagonal*)
- two diagonal lines, crossing each other (*cross*)
- a jagged line, running horizontally (*zig zag*)
- a wavy line, running horizontally (*wave*)

\(^2\)Largely due to the COVID-19 pandemic.
• several lines passing back and forth, horizontally (scratch)

It should be noted that the proposed list is not exhaustive, and other categorisations could be applied. However, we chose to focus on this arrangement as it summarises the strokes that, based on personal observations, are mainly used to strike through words.

In addition to supporting different stroke types, the new method should generate strokes with a considerable degree of variability, i.e. resulting in differing appearances. Generated strokes within a type category should be recognisable as belonging to the same group but not be exactly the same. This for example entails variations in stroke length, slant and, where applicable, density.

Figure 3.2. One example per strikethrough stroke type, considered in this work. a) Single, b) Double, c) Diagonal, d) Cross, e) Zig Zag, f) Wave, g) Scratch. All word images stem from the Dracula\textit{real} [35] (cf. section 3.3.2) dataset and show examples of synthetic strikethrough, generated using our approach.

General Assumptions
Our strikethrough generation method is developed with the concrete aim of providing data for a strikethrough removal approach that can be applied to relevant words in a given text. We therefore assume that the data has already been segmented and that our generator produces individual words, based on single-word inputs.

In addition to this, we assume that the input is a greyscale image. At the time of writing, greyscale conversion is a commonly-used preprocessing step for text recognition (cf. e.g. [21, 45]). It is therefore reasonable to assume that the word images already exist in this colourspace and thus should also be output in the same.

Our third assumption is founded on text preprocessing methods, as well as being of practical nature for the generation process. Here, we assume that the background of a given input image is noise-free, i.e. of a single, solid colour. This will for example make it easier to obtain information related to the actual word, without having to consider any disturbances due to background noise.
As a last general assumption, we limit the range of possible writing tools to those that exhibit a comparably stable stroke width, such as modern pens and pencils. We do not include tools that may result in considerable variations in stroke width, such as quills.

**General Observations**

Prior to implementing our generation approach, we inspected various word and strikethrough images and made a number of personal observations that we incorporated into the method development\(^3\).

Firstly, simple strikethrough strokes, i.e. mainly single and double lines, appear to be largely focused on the core of a word. Other stroke types may considerably extend beyond the core, also covering ascenders and descenders. Generally, and regardless of stroke type, there seems to be a tendency to cover as much of a word as possible to emphasise that this word should be deleted.

Secondly, a large portion of strikethrough is applied immediately during the original writing process, i.e. by the same writer and using the same tool. This makes the word and strikethrough strokes visually indistinguishable. Naturally, time delays may occur, leading to a change in writer or tool, however these can often be attributed to specific types of documents, such as population registers, where the strikethrough can for example indicate a change of address.

As a third observation we note that the intensity varies across the stroke profile for many of the commonly used writing tools. These variations can stem both from the imaging process, as well as being intrinsic to the specific writing tool.

Our final observation pertains to areas where two or more strokes overlap. Here, a change of intensity, can often be observed. The extent of this change largely depends on the type of writing tool, and to some some degree, on the original stroke colour.

Figure 3.3 shows examples of words written with different tools, demonstrating the colour and intensity variations along the stroke trajectory, as well as at junction points.

**Requirements and Assumptions Derived From General Observations**

In addition to the general requirements and assumptions, we derive a number of considerations based on the discussed observations. Firstly, we require that the generated strokes cover a large portion of the word core. Strokes of types *diagonal, cross, zig zag, wave* and *scratch* should also cover portions of ascenders or descenders, where present. Besides this, we assume that a given word has always been struck-through almost immediately, i.e. by the same writer and using the same tool. We do not consider cases with a larger time

---

\(^3\)The observations are primarily based on Latin script and may not generalise to other types of writing.
Figure 3.3. Examples of stroke colour variation across the stroke profile and at junction points. Writing tools: a) fountain pen, b) ballpoint pen, c) pencil, d) gel pen, e) felt pen.
delay, which could lead to a change in writer or writing implement. In general, the generated strikethrough strokes should therefore be visually similar to the input word strokes. This requirement also entails similarities with respect to intensity variations across the stroke profile and changes at junction points. For the latter, we assume that overlapping strokes are darker than single-layer strokes, as additional pigments are deposited.\(^4\)

### 3.2.2 General Approach

Our general stroke generation approach assumes a greyscale word image and the desired stroke type as input. The implementation can be separated into three main steps that are presented in the following.

#### Feature Extraction

In the initial step, various features are extracted from the provided input word image. We firstly calculate the average ink intensity from the provided greyscale input. Afterwards, the word image is binarised, using Otsu’s method [62]. Based on the binarised image, the stroke width and core region are extracted (cf. Figure 3.4 a)). The former is obtained by calculating the median of the horizontal and vertical ink run lengths. For the subsequent generation steps, the stroke width is scaled by a (configurable) factor of 0.75 to account for later deformations, including slight blurring. The core region is extracted by identifying the largest connected region of the thresholded horizontal black run profile, as proposed by Papandreou and Gatos [64]. Lastly, a distance map is calculated, as shown in Figure 3.4 b). In order to prevent artefacts in cases where word strokes touch the image boundary, the binary image is padded with a small margin, which is cropped from the resulting distance map.

#### Strikethrough Stroke Generation

Firstly, a number of control parameters are sampled, based on the word image size and the position of the core region. The number and type of control parameters varies, depending on the stroke type that is being generated. In the example case of a single line, the start and end position are sampled such that the y-coordinates lie within the core region and have a maximum difference of 10px. The x-coordinates should have a distance of at least 75% of the image width and the starting x-position should lie within the first 10% of the image columns. The obtained stroke points and the previously calculated stroke width are used to draw a solid black line on a white canvas (cf. Figure 3.4 c)). Similar to the previous step, the distance map of the generated stroke is calculated, as shown in Figure 3.4 d). Finally, the greyscale word

\(^4\)This does not hold for highly pigmented white pens on dark paper, where additional layers would make the stroke lighter. However, as this is a rare edge-case, and can likely be addressed by inverting the image beforehand, we exclude it from our considerations.
image, its distance map, and the stroke distance map are used to texturise the stroke profile. For this, all word pixels with a given distance are identified and their intensity values collected in a pool. This collection of intensities is then used to sample one intensity value for each of the stroke pixels that share the same distance as the respective pool. One such result is shown in Figure 3.5 a).

![Figure 3.4](image)

Figure 3.4. a) Input image with core region annotation; b) distance map of the binarised input image – distances larger than six have been combined into one level, marked in blue, for visualisation purposes; c) generated binary line profile; d) distance map of the binary line profile (maximum distance = 6).

**Strikethrough Application**

Before finally combining the generated stroke with the greyscale word image, we deform the former slightly, by applying a small Elastic Transformation [79]. This step is performed to increase the resemblance to a stroke created from human movements. Finally, as last step of our generation pipeline, the
A strikethrough stroke is superimposed onto the greyscale word image. As outlined in the discussion of requirements and assumptions (cf. subsection 3.2.1), we observed a change in intensity at points where a strikethrough stroke crosses a word stroke. We model this in our synthetic stroke generation by adapting the intensity of junction pixels as follows:

\[
\text{new\_pixel\_intensity} = \text{original\_word\_intensity} + \lambda \times \text{stroke\_intensity}
\]

(3.1)

where:

\[
\lambda = \frac{\text{average\_stroke\_intensity}}{\text{max\_stroke\_intensity}}
\]

(3.2)

calculated based on the intensities of the original word image. Lastly, the newly created image is smoothed with a Gaussian blur (3x3 mask, \(\sigma = 0.8\)) in the areas that denote the original strikethrough stroke and, where applicable, intensities are clipped to \([0, 255]\). The resulting image is shown in Figure 3.6.
3.2.3 Perspectives

Our developed approach presents a lightweight method for automatically generating struck-through words, and does not require any human interaction. One aspect that has not been integrated is the simulation of human movement, for example as proposed by Likforman-Sulem and Vinciarelli [51]. This research direction was not investigated further for ease of implementation at the time of development. However, the integration of such an approach and its impact on the strikethrough removal performance would be an interesting case study.

Another avenue for future work is the exploration of other opportunities to automatically generate struck-through words. To the best of our knowledge, no deep-learning-based attempt, for example using generative adversarial networks (GANs), has been made at the time of writing.

3.3 Datasets

As discussed initially, the Lindgren manuscripts served as primary motivation for investigating strikethrough processing. However, at the time of writing of Papers I and II, the digitisation and annotation of these documents was still in progress. We therefore focused our work on other types of handwriting, concretely Latin script, with the aim of later transferring any findings and models to the Lindgren corpus. In this regard, three datasets in Latin script were created to form the basis for our experiments. This curating step was primarily necessary as the datasets used in prior strikethrough-related works were not available at all or could only be obtained without the strikethrough-free ground-truth images. The three datasets are introduced in more detailed in the following sections. Furthermore a brief summary is presented in Table 3.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Synthetic</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
<th>Multi-Writer</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM&lt;sub&gt;synth&lt;/sub&gt;</td>
<td>yes</td>
<td>3066</td>
<td>273</td>
<td>819</td>
<td>Yes</td>
</tr>
<tr>
<td>Dracula&lt;sub&gt;real&lt;/sub&gt;</td>
<td>no</td>
<td>126</td>
<td>126</td>
<td>378</td>
<td>No</td>
</tr>
<tr>
<td>Dracula&lt;sub&gt;synth&lt;/sub&gt;</td>
<td>yes</td>
<td>5 × 126</td>
<td>N/A</td>
<td>N/A</td>
<td>No</td>
</tr>
</tbody>
</table>

3.3.1 IAM<sub>synth</sub>

The first dataset that we developed as part of Paper I is based on the IAM database [55]. As the name IAM<sub>synth</sub> indicates, it was created using our synthetic strikethrough generation approach, presented in section 3.2. Concretely, we applied the following steps to create the altered images:
1. from a given page, a word is sampled (without replacement), with width and height in the range [60,600] px\(^5\)
2. for each word image, the background is removed, using [83]
3. the strikethrough generation is applied, using default parameters
4. the original, clean image and its altered counterpart are stored

Using this approach, we obtained a total of 3066 train, 273 validation and 819 test image pairs. The sampling of clean word images is based on the original IAM writer identification splits, which ensure that a given writer is only present in one of the three datasplits. Apart from this, and the previously mentioned size restrictions, we do not limit the word image selection in any form, i.e. we do not control for statistics like word length, stroke width or textual content. Figure 3.7 shows one generated example per stroke type from the IAM\(_{synth}\) training set.

![Figure 3.7. One example per stroke type from the IAM\(_{synth}\) dataset. a) Single, b) Double, c) Diagonal, d) Cross, e) Zigzag, f) Wave, g) Scratch.](image)

3.3.2 Dracula\(_{real}\)

In order to not have to rely exclusively on the synthetic strikethrough dataset, IAM\(_{synth}\), we also created a small, genuine strikethrough corpus in Paper I. The contained words and strikethrough strokes stem from a single writer, and cover an excerpt from Bram Stoker’s novel *Dracula*\(^6\). We created this dataset by scanning the clean, handwritten pages (white printer paper, 80g/m\(^2\), blue ballpoint pen), applying word-level strikethrough to the original with the same pen, and then scanning the altered document. The two versions of each page, clean and struck-through, were aligned using the registration method proposed by Öfverstedt et al. [92], before segmenting the individual words. Afterwards,

\(^5\)This size limitation was applied to prevent artefacts during the strikethrough generation process.

\(^6\)Chosen primarily because it has been in the public domain for a considerable amount of time, thus dispelling any copyright concerns.
each obtained word image pair was manually annotated with the respective stroke type. Figure 3.8 shows one sample per stroke type from the Dracula\textsubscript{real} dataset.

The 756 obtained image pairs were shuffled and split into portions for training, validation and testing, ensuring an even distribution of the seven stroke-types in each split. We did not balance the datasplits with respect to any other statistic, such as the image content (transcription, word length) or size. As can be seen in Table 3.1, the training and validation set consist of 126 images each, i.e. 18 samples per stroke-type, while the test set contains 378 images, i.e. 54 samples per stroke-type. In addition to this, the training set contains an additional 126 images for which the struck-through images are included in the dataset but not considered in this splitting. For this portion, only the clean images are considered to be of relevance, particularly in the context of training CycleGANs, which will be discussed in section 3.5.

![Figure 3.8. One example per stroke type from the Dracula\textsubscript{real} dataset. a) Single, b) Double, c) Diagonal, d) Cross, e) Zigzag, f) Wave, g) Scratch.](image)

3.3.3 Dracula\textsubscript{synth}

The third strikethrough-removal dataset is based on the clean images from Dracula\textsubscript{real} and was created using our proposed strikethrough generation approach. This dataset was primarily created to bridge the gap between the two others and to emulate the Lindgren use case, i.e. a single-writer corpus for which we want to remove strikethrough strokes but don’t have access to a suitable paired dataset.

Overall, we generated five training partitions, each containing one struck-through example for each word in the Dracula\textsubscript{real} training set. In our experiments in Paper II, we consider these partitions both individually, and as one large dataset that is five times the size of the training set from Dracula\textsubscript{real}.

Figure 3.2 shows some examples from the Dracula\textsubscript{synth} dataset.
3.4 Strikethrough Identification and Classification

During the experiment design for the strikethrough-removal, we took the decision to focus on word-based processing, both in regard to strikethrough-stroke generation and removal. If we consider a manuscript that has already been segmented into words, the next step in the strikethrough-removal pipeline would therefore be to separate the words that need to be cleaned from those that already are clean. In addition to this, depending on the strikethrough-removal method, one might also want to separate the affected words into different stroke types. To some degree, the stroke type classification could also be used to identify words that are more or less severely affected by strikethrough. One can argue that a word, struck-through with a single line is generally easier to decipher for a human than, for example, one affected by zigzag (cf. e.g. Figure 3.8 a) and f)). Hence one might want to devote more attention to cleaning the latter than the former. In Paper I, we briefly explore both of these filtering tasks, using the DenseNet121 [40] architecture, which was state-of-the-art at the time.

For the first task of identifying whether a given word is clean or struck-through, a two-class DenseNet121 [40], trained on IAM\textsubscript{synth} achieves an $F_1$ score of 0.9899 (Standard Deviation (SD) = 0.0031) for the test split. Evaluating the same models on the test portion of Dracula\textsubscript{real} yields an average $F_1$ score of 0.7759 (SD = 0.0243), indicating that the models trained on the synthetic data do not fully generalise to the unseen dataset.

Regarding the second task of classifying the type of strikethrough present in a given image, we trained a seven-class DenseNet121 [40] on the struck words from IAM\textsubscript{synth}. For these experiments, we assume the seven types of strikethrough strokes, introduced in subsection 3.2.1, and a perfect identification of struck-through words in the preceding step. Similar to the previous task, a high macro $F_1$ score of 0.9238 (SD = 0.0060) is obtained for the IAM\textsubscript{synth} test set, while generalisation to the Dracula\textsubscript{real} test set is noticeably lower at a macro $F_1$ score of 0.6170 (SD = 0.0405). Considering the concrete misclassifications, we note that a considerable amount of synthetic wave strokes is being identified as zig zag and vice-versa. A reasonable explanation for this result can be found in the generation process of the two stroke types. In both cases, the same method is used to generate the control points, which are then represented as the two stroke types by varying the interpolation method. A similar outcome can be observed for the wave and zig zag strokes of the Dracula\textsubscript{real} test set. This may both be rooted in the classification issues observed for the IAM\textsubscript{synth} dataset, as well as in the way these strokes are expressed in genuine handwriting. Inspecting the sample shown in Figure 3.9, it can be observed that when drawn sloppily, for example due to an increased writing speed, the distinction between wave and zigzag decreases, as some corners may become softer, i.e. slightly rounded, and wavy strokes more pointed. Besides this,
a considerable amount of genuine *cross* strokes are misclassified as *diagonal* ones. However, we have not been able to identify a concrete reason for this.

In conclusion, we have demonstrated that the DenseNet121 architecture is suitable to filter and classify synthetic strikethrough. A reasonable performance can be obtained when using the models, trained on synthetic data, to classify genuine data. However, fine-tuning the existing models, or re-training them from scratch, on the genuine data is expected to increase the classification performances on Dracula$_{\text{real}}$.

### 3.5 Unpaired Image to Image Translation (Paper I)

The initial approach that we took to remove strikethrough from handwritten words was to explore the use of unpaired image to image translation, concretely in the form of CycleGANs [91] (cf. section 2.5.4). This group of methods is of particular interest to strikethrough removal because it does not require the availability of paired data, which, as discussed extensively in earlier sections, can be challenging to acquire.

One concern that we encountered regarding the use of a standard CycleGAN for strikethrough removal is that, in its original configuration, a considerable one-to-one mapping exists between the two image domains. This is crucial for the training process, as the cycle-consistency loss is based on the idea that an image can be mapped almost perfectly from one domain into the other and back, and still be of considerable similarity to the original input. While this assumption holds if we consider the cycle that departs from a clean image, it is not guaranteed for the opposite direction. Given a clean word, a large variety of strikethrough strokes are plausible, making the strikethrough stroke generation a one-to-many relationship.
In order to mitigate this concern, we considered different approaches for guiding the training process in a way that resolves, or reduces, the impact of the one-to-many relationship between a clean image and its potential strike-through strokes. We chose to explore attribute-guided CycleGANs, a method proposed by Lu et al. [52], who integrate additional domain knowledge by expanding the input images with data channels, encoding relevant features. Lu et al. furthermore introduce a pre-trained auxiliary discriminator that regularises the generator [52]. Based on prior experiments, we decided to integrate attribute-guided aspects into our experiments in several ways. Firstly, we expanded the clean image inputs with seven additional channels, one per stroke type, effectively resulting in a channel-wise one-hot encoding of the target stroke type. We did not integrate any information into the struck inputs, as the target output is well-defined for these. Besides this input expansion, we adapted the proposed auxiliary discriminator to employ the stroke classification network that we introduced in section 3.4. Following the original attribute-guided approach, we remove the final layer of the pre-trained auxiliary network and use it as feature extractor, based on which the auxiliary loss is computed.

Overall, we trained and examined all four combinations of the original CycleGAN with and without the addition of attribute-guiding features and the auxiliary discriminator. All three modifications yielded improvements with respect to the $F_1$ score and RMSE, as compared to the performance of the original CycleGAN. Combining both, the additional features and the auxiliary discriminator, resulted in the largest increase in performance, with an overall $F_1$ score of 0.8172 (SD = 0.0145) and RMSE of 0.0833 (SD = 0.0045) for IAM$_{synth}$, and 0.7376 (SD = 0.0107), respectively 0.0576 (SD = 0.0013) for Dracula$_{real}$.

When comparing the cleaning performances between the different stroke types, considerable differences can be noted. Generally, all four model configurations obtain the best scores for the stroke types single and double, on both datasets. This is to be expected, as both strokes only cover small portions of the word and in a manner that is structurally different from most word strokes. The worst scores for all models and both datasets are obtained for scratch strokes. In many cases, this type heavily affects and occludes words, thereby considerably increasing the challenge of obtaining the original clean image. As indicated above, a drop in performance can be observed when comparing the results from IAM$_{synth}$ with those obtained for Dracula$_{real}$.

Lastly, we present a selection of images, demonstrating the qualitative cleaning performance of the different models, for both datasets. The images were handpicked with the aim of demonstrating the range of cleaning capabilities, from convincing results in Figure 3.10 to failure cases in Figure 3.11. Considering the former, it can be noted that some residual stroke remain, especially for the original CycleGAN and the attribute-guided approach without the auxiliary discriminator. For the set of negative examples, it can be observed that
the models either do not remove any of the strokes as at all, or remove large portions of the word alongside the strikethrough. The selection of images also demonstrates the varying levels of challenge between the different stroke types, and their impact on the original word image. Considering for example the fifth and seventh column of Figure 3.11, it is much more challenging to identify which strokes belong to the word, as compared to the seventh column of Figure 3.10.

Figure 3.10. Cherry-picked examples for the four models. First three examples taken from IAM_{synth}, others from Dracula_{real}. Image source: [36].

Figure 3.11. Lemon-picked examples for the four models. First three examples taken from IAM_{synth}, others from Dracula_{real}. Image source: [36].

3.6 Paired Image to Image Translation (Paper II)

One of the main motivations for exploring CycleGANs in Paper I was their suitability for unpaired data scenarios. As discussed, paired strikethrough data can be challenging to obtain, and no such dataset was publicly available before
our work in Paper I. Due to the development of our strikethrough generation approach, and the publication of the related datasets, paired approaches are now feasible. In Paper II, we explored one such approach, concretely paired image to image translation. This group of methods aims to learn the one-to-one transformation of an image from a source to a target domain. In the context of strikethrough removal, the former refers to struck-through images, while the latter domain entails the clean counterparts.

As part of Paper II, we examined four different paired architectures – shown as schematics in Figure 3.12 – as well as the best performing one, the attribute-guided CycleGAN with auxiliary discriminator, from Paper I. The four paired architectures were chosen to explore a variety of model sizes, ranging from the SimpleCNN, with roughly 28,000 parameters, to Generator, with around 1.3 million. Furthermore, the latter was included as additional level of comparison with the CycleGAN, as it corresponds to the architecture that is used for the generators in Paper I, hence the name. Regarding the other two architectures it can be noted that Shallow (ca. 150,000 parameters) was included because it corresponds to the Generator, without the bottleneck layers in the middle, while UNet – in an admittedly very small configuration7 (ca. 180,000 parameters) – was included due to the popularity of U-Nets [74] at the time.

**Figure 3.12.** Schematic overview of the four architectures considered in the paired image-to-image translation setting. Sizes are not perfectly to scale, however Shallow and UNet are of similar size, while SimpleCNN and Generator are considerably smaller, respectively larger. Image source: [15].

7Larger configurations were explored in preliminary experiments but fell short of the presented one.
Using the three datasets, IAM\textsubscript{synth}, Dracula\textsubscript{real} and Dracula\textsubscript{synth}, we examined three different data scenarios to determine how these affect the model performances. Firstly, we trained the models on IAM\textsubscript{synth} and evaluated it on the same dataset, as well as on Dracula\textsubscript{real}. This experiment was primarily undertaken to establish a basis for comparison with Paper I. Considering the evaluation on IAM\textsubscript{synth}, all paired approaches outperform the CycleGAN by 7 to 17 percentage points (pp), with respect to the $F_1$ score. The Generator architecture, the largest of the paired approaches, achieves the best results with an $F_1$ score of 0.9697 (SD = 0.0012) and RMSE of 0.0237 (SD = 0.0016). Evaluating these models on Dracula\textsubscript{real} results in a similar outcome as in Paper I, with all of them performing considerably worse on the unseen data. Here, the difference in performance between architectures has reduced considerably, however the Generator still outperforms all others.

In the second experiment setting, all models are trained on the individual partitions of Dracula\textsubscript{synth}, and evaluated on Dracula\textsubscript{real}. Despite the much lower amount of data – 126 images, as compared to 3066 for IAM\textsubscript{synth} – the performance of all image-to-image translation models has increased slightly, indicating that they benefit from a dataset that is closer to the target domain than IAM\textsubscript{synth}. In contrast to this, the CycleGAN’s performance has decreased considerably, by approximately 21 pp for the $F_1$ score, which can be attributed to the small amount of training data.

Based on the improvement for the image-to-image translation models, trained on the small Dracula\textsubscript{synth} dataset, the question arose whether an increase in training data would further improve the results. In the third experiment setting, we therefore combined the five Dracula\textsubscript{synth} partitions into one large set (630 images) and trained all five models accordingly. Again, the performance of the image-to-image translation models improved slightly, by about 2 pp for the $F_1$ score. The performance of the CycleGAN increased considerably, by about 17 pp, supporting the aforementioned reliance on a larger training set size. Regardless, the latter architecture is still outperformed by the paired approaches by approximately 7 to 13 pp.

As before, we present a number of handpicked positive and negative examples in Figure 3.13 and Figure 3.14, respectively. Both figures show mean greyscale images taken from Dracula\textsubscript{real} and aggregated over 30 repeated runs per architecture, training on the combined partitions of Dracula\textsubscript{synth}. Similar to the results in Paper I, we observe that the wave and zigzag strokes pose considerable challenges to the models, while single and double strokes are cleaned with satisfying performance.

Overall, several conclusions can be drawn, based on the results, obtained from these three experiments. Firstly, as indicated above, the CycleGAN architecture requires larger amounts of data, as indicated by the performance difference on the three training set configurations. Secondly, under the given data restrictions and for the task at hand, paired approaches perform better than CycleGANs. In an application setting where paired data is available, or
can be created, these approaches should therefore be explored first. Regarding the different paired architectures, we observed the highest performances for the largest model. It should however be noted that even the smallest one, SimpleCNN, at two percent of the size of the Generator, displayed reasonable performances. One contributing factor in this regard is likely the supervised setting under which the paired approaches were trained, allowing for a more direct impact of errors than in the CycleGAN setting.

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>pretended tongue further comforting and</th>
</tr>
</thead>
<tbody>
<tr>
<td>Struck Input</td>
<td>pretended tongue further comforting and</td>
</tr>
<tr>
<td>SimpleCNN</td>
<td>pretended tongue further comforting and</td>
</tr>
<tr>
<td>Shallow</td>
<td>pretended tongue further comforting and</td>
</tr>
<tr>
<td>UNet</td>
<td>pretended tongue further comforting and</td>
</tr>
<tr>
<td>Generator</td>
<td>pretended tongue further comforting and</td>
</tr>
<tr>
<td>CycleGAN</td>
<td>pretended tongue further comforting and</td>
</tr>
</tbody>
</table>

*Figure 3.13. Cherry-picked examples for the five models from Paper II. All examples taken from the Dracula_{real} test split. Results are shown as mean greyscale images, summarised over 30 repeated training runs for each architecture. Image source: [15].*

### 3.7 Conclusions and Perspectives

This chapter has summarised the work that was performed in Papers I and II. The primary conclusions are:

- the Densenet121 [40] architecture is suitable for filtering struck-through words from clean ones and separating them into different stroke types
- in scenarios where training data is limited, paired image-to-image translation approaches are more suited than CycleGANs for strikethrough-removal
- in settings where computational resources are sparse, smaller paired image-to-image translations approaches may be a viable option, as these have demonstrated considerable cleaning performances, albeit slightly lower than the larger investigated models
- our synthetic strikethrough generation approach is a suitable option to create artificial training data for the identification, classification and clean-
ing of strikethrough, however results indicate that the implementation for \textit{zigzag} and \textit{wave} strokes should be revisited

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|}
\hline
Ground Truth & all the \textit{Came} simply he & & \\
\hline
Struck Input & \textit{Came} \textit{Came} simply he & & \\
\hline
SimpleCNN & \textit{Came} \textit{Came} simply he & & \\
\hline
Shallow & \textit{Came} \textit{Came} simply he & & \\
\hline
UNet & \textit{Came} \textit{Came} simply he & & \\
\hline
Generator & \textit{Came} \textit{Came} simply he & & \\
\hline
CycleGAN & \textit{Came} \textit{Came} simply he & & \\
\hline
\end{tabular}
\caption{Lemon-picked examples for the five models from Paper II. All examples taken from the Dracula\textsubscript{real} test split. Results are shown as mean greyscale images, summarised over 30 repeated training runs for each architecture. Image source: [15].}
\end{table}

Regarding the aspect of computational resources it should be mentioned that the training of CycleGANs entails at least four different models, two generators and two discriminators. However, as we are only interested in the capabilities of the generator that processes struck-through words, the other three models are only relevant during training and are discarded afterwards, thus incurring a large computational overhead. In the interest of Green AI and considering the low performance, compared to “cheaper” models, the use of CycleGANs in real application scenarios cannot be recommended at the time of writing.

Upon personal reflection on the work that has been performed in regard to strikethrough processing, there are number of topics that I would like to address in the future. Firstly, all of the models have been trained exclusively on synthetic data. Although manually preparing suitable training data is prohibitively costly, it would be interesting to explore this direction to be able to gain more insights with respect to the differences and similarities between genuine and synthetic data. The already existing Dracula\textsubscript{real} dataset could have been used for such an investigation, however, at the time I did not follow this line of research, because it does not correspond to realistic data settings, for example when handling strikethrough in an archival context. In this regard it would also be interesting to explore the data generation approach that superim-
poses genuine strokes onto words, as suggested by Poddar et al. [68]. Lastly, this work was originally developed to be applied to the data in *The Astrid Lindgren Code* project, in order to make struck-through texts more accessible to humans and machines. The implementation of strikethrough approaches for handwritten stenography is under preparation.

Finally, considering the aspects of reuse and reproducibility, two related works have to be mentioned. Firstly, Nigam et al. [61] report the use of our stroke generation method in their work. Unfortunately, their dataset and code are only available upon request and not public. However given that the code for our generation method is publicly available, it should be easier to reproduce their work, even if the original data cannot be obtained.

Secondly, Basting et al. three Master’s students from Delft University of Technology, independently explored reproducibility aspects for Papers I and II [6]. They do not comment on whether they consider the reproduction to be successful or not. However, overall, the independently obtained results are close to the original performance. Besides reporting their results, the authors commented on some difficulties in understanding the code, especially for obtaining the test performance. This has prompted some changes in the general way in which I structure my code and is for example reflected in the software artefacts for Papers III and IV.
4. Handwritten Stenography Recognition
(Paper III and IV)

Before the inception of the Astrid Lindgren Code project (2020-2023), the prevailing opinion was that Lindgren’s manuscripts were near impossible to decipher [60]. Since then, a group of dedicated volunteers has demonstrated, that it is indeed possible for other stenographers to read these documents [4]. This directed transliteration effort has enabled the exploration of handwritten text recognition approaches for Swedish stenography.

Although various different stenography systems exist, such as Pitman’s and Gregg’s for English, Gabelsberger for German and Melin for Swedish, the field of stenography recognition is small. One of the earliest works that we could identify was published in 1985 by Brooks and Newell, and approaches stenography recognition from the level of primitives, i.e. small strokes, like loops and hooks [12]. Since then, Leedham and colleagues have explored various approaches and research questions, primarily focusing on classical methods in the context of online Pitman’s shorthand [38, 39, 48, 49, 53, 72]. Besides this, a few works, employing neural networks, have been published since the early 2010s. Here, the primary focus has been to classify selected terms for Pitman’s [58] and Gregg’s [63], and to transliterate individual words for the latter system [89]. To the best of our knowledge, no prior work has attempted to perform extensive, computerised transliterations of manuscripts written in any stenographic system.

In this chapter, Papers III and IV are summarised, which focus on HTR for stenography, or Handwritten Stenography Recognition (HSR). Before presenting methodological details, the LION dataset is introduced, which forms the basis for all of our experiments.

4.1 The LION Dataset

One of the contributions of Paper IV is the introduction of the LION dataset, consisting of portions taken from Astrid Lindgren’s manuscripts, written in Melin’s stenographic system. The dataset’s name is both a tribute to Astrid Lindgren (LI), and The Brothers Lionheart (Swedish: Bröderna Lejonhjärta), which forms the bulk of the dataset and is one of the writer’s famous children’s books. In addition to The Brothers Lionheart, the dataset also contains excerpts from another children’s story, Emil of Lönneberga (Swedish: Emil i
Lönneberga), and portions of the biographic and autobiographic texts, *Samuel August from Sevedstorp and Hanna in Hult* (Swedish: Samuel August från Sevedstorp och Hanna i Hult) and *On our grove* (Swedish: Om vår hâge). Besides this, some of the notepads also contain additional pages, with personal notes, letter drafts, etc., which are excluded from this project for privacy reasons.

The annotated data was produced by combining the crowdsourced transcriptions, which contain line and page break indicators, with the segmented line images. Overall, the dataset spans eight of the 670 archived notepads, and consists of 2900 lines of handwriting, across 198 pages. An example for a page from the notepads was shown in an earlier chapter, cf. Figure 2.8. In the following sections we briefly introduce selected considerations, and aspects of the dataset.

4.1.1 Data Splitting

As indicated, the different literary works are not equally represented in the LION dataset. Figure 4.1 gives an overview over the page count per notepad and text, with *Lionheart-[1-6]* referring to the first six chapters of *The Brothers Lionheart*, *Emil-[1-2]* to two separate portions of *Emil from Lönneberga*, *Parents-1* to *Samuel August from Sevedstorp and Hanna in Hult* and *Autobiography-1* to *On our grove*. Regarding the notepads, covering the *The Brothers Lionheart*, it should be noted that some chapters are present in several revisions, for example the first and second one. Furthermore, three transitional pages exist, indicated as “Lionheart-1|Lionheart-2”, and “Lionheart-

---

1Samuel August and Hanna are the first names of Lindgren’s parents.
Since this dataset is intended to be used for deep learning and similar experimental settings, a splitting into training, validation and test sets was prepared. Considering the distribution of pages and textual contents, a variety of splittings are conceivable. After careful consideration, we decided to split the dataset into two portions. The first, consisting entirely of pages from chapters one to three, five and six, of *The Brothers Lionheart*, forms the basis for 5-fold cross-validation splits that are randomly sampled at line-level. The rest of the pages are split further, to create two test portions. Firstly, an in-domain test set, containing all lines from chapter four of *The Brothers Lionheart*, aptly named Test-LH. And secondly, an out-of-vocabulary test set, thus named Test-OOV, encompassing all lines from Emil-[1-2], Autobiography-1 and Parents-1. The datasplits were arranged in this way with the aim of leveraging the large amount of already available transliterations for *The Brothers Lionheart* for training and validation, and to ascertain how well a given model performs on the task of transliterating text from the same domain, as well as its generalisation to texts that are written in other linguistic styles.

To demonstrate the linguistic variation between the splits, we performed a brief quantitative analysis of the contents. For this, we firstly removed the stopwords, based on the list of Swedish stopwords provided by NLTK [8]. Afterwards, we calculated the pairwise cosine similarities between the Term Frequency – Inverse Document Frequency (TF-IDF) [43] vectors for the three documents – the entirety of the cross-validation set (indicated as “Cross-Val”), Test-LH and Test-OOV. Figure 4.2 summarises this comparison, demonstrating that there is a considerable similarity between the portions that are based on *The Brothers Lionheart* (0.81), and much less between the former two and the out-of-vocabulary split, Test-OOV (0.16, resp. 0.19).

### 4.1.2 Editorial Marks

One central aspect of Lindgren’s manuscripts is the large amount of editorial marks. The pages in the LION dataset contain a variety of alterations, in the form of strikethrough (cf. Figure 4.3) and additions (cf. Figure 4.4). Here, strikethrough is generally used to indicate deletions, while additions, typically written above, or occasionally below, the original line, indicate amendments, made by the author. Overall, both of these alterations indicate revisions and are a product of Lindgren’s editorial process, and thus of interest to the *Astrid Lindgren Code* project.

Approaches for cleaning struck-through words have been discussed in detail in chapter 3. Generally, additions do not present the same issues of potentially
Pairwise Cosine Similarity (TF-IDF)

<table>
<thead>
<tr>
<th></th>
<th>Cross-Val</th>
<th>Test-LH</th>
<th>Test-OOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Val</td>
<td>1</td>
<td>0.81</td>
<td>0.19</td>
</tr>
<tr>
<td>Test-LH</td>
<td>0.81</td>
<td>1</td>
<td>0.16</td>
</tr>
<tr>
<td>Test-OOV</td>
<td>0.19</td>
<td>0.16</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 4.2.** Pairwise cosine similarity of the TF-IDF vectors for the cross-validation (“Cross-Val”) portion of the LION dataset and the two different test sets. A higher value (range: [0, 1]) indicates a larger similarity between the respective sets.

unreadable words. They do, however, introduce challenges related to resolving the reading order of words in a given text line. The mapping between word positions in the image and the corresponding transliterations is not straightforward, since it requires information about where an addition starts and ends, and where it is supposed to be inserted into the original text line.

As can be seen in Figure 4.5, roughly 20% of the lines are, to a varying degree, affected by one of these two types of editorial marks. The other two shown categories, clean and missing, refer to lines without any alterations, and lines for which the transliterations are incomplete, respectively. We chose to include the two line types, bearing editorial marks, in the dataset with the aim of giving researchers the opportunity to explore questions related to the processing and recognition of such disturbances. Missing lines are included for completeness but not considered in any of the works presented in the rest of this chapter.

For each of the three datasplits, training, validation and testing, we provide subsets, which only contain specific types of lines or combinations thereof, for example only encompassing the clean training lines, or a combination of the clean and struck validation lines. Table 4.1 summarises the distribution of line types per datasplit.

---

2An exception to this are instances, in which a writer is attempting to fit an addition into a tight space, resulting in overlaps or heavily deformed (squeezed) characters.
Figure 4.3. Examples for lines containing varying amounts and styles of strikethrough.

Figure 4.4. Examples for lines containing additions, individually (top) and in combination with struck-through words (bottom).

Table 4.1. Line counts per datasplit and line type

<table>
<thead>
<tr>
<th>Datasplit</th>
<th>Clean</th>
<th>Struck</th>
<th>Added</th>
<th>Split Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>1224</td>
<td>196</td>
<td>200</td>
<td>1620</td>
</tr>
<tr>
<td>Validation</td>
<td>306</td>
<td>49</td>
<td>50</td>
<td>405</td>
</tr>
<tr>
<td>Test-LH</td>
<td>474</td>
<td>31</td>
<td>24</td>
<td>529</td>
</tr>
<tr>
<td>Test-OOV</td>
<td>191</td>
<td>31</td>
<td>34</td>
<td>256</td>
</tr>
<tr>
<td>Type Total</td>
<td>2195</td>
<td>307</td>
<td>308</td>
<td>2810</td>
</tr>
</tbody>
</table>

Line Distribution Across All Pages

Figure 4.5. Distribution of lines in the LION dataset across the four different types.
4.2 Impact of Augmentations (Paper III)

Paper III presents the results of our HSR precursor study, which investigates the impact of commonly-used HTR augmentation techniques in a stenographic context. As discussed earlier, data augmentations are frequently used in low-resource settings, which the LION dataset is an example of. In HTR and DIP, rotations, scaling, shifting and shearing, as well as morphological operations, are often used [21, 46, 71, 73, 84, 86, 87]. These techniques introduce variations that occur naturally in handwriting, even for one and the same writer.

While these variations can also occur in stenography, some of the transformations coincide with the differences between some character pairs, such as small rotations for “a” and “e” or scale for “o” and “å” (cf. Figure 2.4). One of the main motivations for this study is therefore the question whether such augmentations can be used for stenography or whether they may lead to confusions between characters, thus leading to a decrease in recognition performance. In order to investigate this question, we examined eleven types of augmentation operations, with a total of 22 parameter variations, summarised in Table 4.2. Most of these configurations were chosen based on prior works in HTR and DIP, while the last category (row 20-24 in Table 4.2) was added to explore a selection of techniques that alter the image without affecting the shape or scale of different symbols. For each of the selected augmentation configurations, a CTC-based architecture (Gated-CNN-BGRU [21]) was trained, individually applying the respective transformation to each training image with a rate of 0.5 per epoch. The results were compared against those, obtained from an augmentation-free baseline using the same architecture.

Overall, we conclude that several of the typical HTR and DIP augmentation techniques (small rotations, shifting, shearing, elastic transformations and scaling) can be applied to stenography, without having identifiable adverse effects. No indications could be found regarding the concern that these transformations lead to confusions between symbols, which may, in part, be attributable to the fact, that the augmentations were applied at a line, instead of a symbol-level, thus preserving much of a given character’s visual context within a word.

4.3 Integrating Stenographic Theory (Paper IV)

In Paper IV, we begin by introducing the LION dataset and establishing a baseline for HSR, using a state-of-the-art, CTC-based HTR architecture, in the form of a slightly modified version of the Gated-CNN-BGRU [21]. This baseline setup achieves a mean CER of 29.81% (SD = 0.92) and WER of 55.14% (SD = 1.26). For reference, applying the same training and evaluation protocols for the similarly-sized dataset Saint Gall [24], written in Latin characters, a CER of 6.3%, and WER of 39.65% are obtained, demonstrating the challenging nature of the LION dataset.
Table 4.2. Summary of augmentation configurations. For dilation and erosion, “SE” refers to the structuring element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Augmentation Type</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>rot1.5</td>
<td>Random Rotation</td>
<td>$[-1.5, 1.5]$ degrees</td>
</tr>
<tr>
<td>rot5</td>
<td>Random Rotation</td>
<td>$[5, 5]$ degrees</td>
</tr>
<tr>
<td>rot10</td>
<td>Random Rotation</td>
<td>$[10, 10]$ degrees</td>
</tr>
<tr>
<td>positive</td>
<td>Random Rotation</td>
<td>$[0, 1.5]$ degrees</td>
</tr>
<tr>
<td>negative</td>
<td>Random Rotation</td>
<td>$[-1.5, 0]$ degrees</td>
</tr>
<tr>
<td>rot+2</td>
<td>Fixed Rotation</td>
<td>2 degrees</td>
</tr>
<tr>
<td>rot-2</td>
<td>Fixed Rotation</td>
<td>-2 degrees</td>
</tr>
<tr>
<td>square-dilation</td>
<td>Random Dilation</td>
<td>square SE, [1..4] px</td>
</tr>
<tr>
<td>disk-dilation</td>
<td>Random Dilation</td>
<td>disk SE, [1..4] px</td>
</tr>
<tr>
<td>square-erosion</td>
<td>Random Erosion</td>
<td>square SE, [1..3] px</td>
</tr>
<tr>
<td>disk-erosion</td>
<td>Random Erosion</td>
<td>disk SE, [1..3] px</td>
</tr>
<tr>
<td>elastic</td>
<td>Random Elastic Transform. [79]</td>
<td>$\alpha = [16, 20]$, $\sigma = [5, 7]$</td>
</tr>
<tr>
<td>shear</td>
<td>Random Horizontal Shearing</td>
<td>$[-5, 30]$ degrees</td>
</tr>
<tr>
<td>shear30</td>
<td>Random Horizontal Shearing</td>
<td>$[-30, 30]$ degrees</td>
</tr>
<tr>
<td>scale75</td>
<td>Random Scaling</td>
<td>$[0.75, 1]$</td>
</tr>
<tr>
<td>scale95</td>
<td>Random Scaling</td>
<td>$[0.95, 1]$</td>
</tr>
<tr>
<td>shift</td>
<td>Random Shift</td>
<td>$\text{horizontal} = [0, 15]$ and $\text{vertical} = [-3.5, 3.5]$</td>
</tr>
<tr>
<td>mask10</td>
<td>Random Column Masking</td>
<td>10% of columns</td>
</tr>
<tr>
<td>mask40</td>
<td>Random Column Masking</td>
<td>40% of columns</td>
</tr>
<tr>
<td>noise</td>
<td>Gaussian Noise</td>
<td>$\sigma = {0.08, 0.12, 0.18}$</td>
</tr>
<tr>
<td>dropout</td>
<td>Pixel Dropout</td>
<td>$[0, 20]$% of pixels</td>
</tr>
<tr>
<td>blur</td>
<td>Gaussian Blur</td>
<td>kernel=5, $\sigma = [0.1, 2]$</td>
</tr>
</tbody>
</table>

For the HSR baseline above, and in Paper III, the automatic transliteration of stenography has been treated as a regular HTR task. This approach however, does not take any of the characteristics into account that set this writing system apart from others, such as Latin. One of these features is, that the symbol set of Melin’s stenography system is considerably larger than the Swedish alphabet\(^3\). Many of the stenographic symbols will therefore result in a transliteration, consisting of two or more Swedish characters. Correspondingly, a given HTR model will have to emit two or more symbols per timestep. While this may be feasible for some of the mappings, it may also result in character omissions for longer target sequences or conflicts during the final CTC decoding step.

In Paper IV, we therefore explore four ways of representing selected stenographic symbols directly, taking inspiration from diplomatic transcriptions. The goal of this transformation, or encoding, step is to increase the one-to-one correspondences between the stenographic and transliterated representations. Below, the four chosen encoding schemes are briefly introduced, followed by a

\(^3\)This is also true for many, if not all, other systems of stenography, such as Gregg’s, Pitman’s and Gabelsberger’s.
4.3.1 Considered Encoding Schemes

Below, the four encoding schemes, titled character shortform, suffix, n-gram and Melin, are briefly introduced. Table 4.3 shows the result of applying each encoding to the sample sentence, given in the first row. All encodings were implemented in a reversible manner, for example by resolving n-grams from largest to smallest, and shortforms before other symbols, in order to ensure that partially overlapping transliterations do not interfere with each other.

Table 4.3. Demonstration of applying the four different encoding schemes on a sample string.

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Character Shortform</th>
<th>Suffix</th>
<th>N-gram</th>
<th>Melin</th>
</tr>
</thead>
<tbody>
<tr>
<td>jag</td>
<td>jag</td>
<td>j</td>
<td>jag</td>
<td>jag</td>
<td>β</td>
</tr>
<tr>
<td>tänkte</td>
<td>tänkte</td>
<td>tänkte</td>
<td>d*</td>
<td>tä&amp;e</td>
<td>γτε</td>
</tr>
<tr>
<td>att</td>
<td>att</td>
<td>att</td>
<td>att</td>
<td>att</td>
<td>att</td>
</tr>
<tr>
<td>det</td>
<td>det</td>
<td>det</td>
<td>var</td>
<td>det</td>
<td>δ</td>
</tr>
<tr>
<td>var</td>
<td>var</td>
<td>var</td>
<td>finare</td>
<td>var</td>
<td>ε</td>
</tr>
<tr>
<td>finare</td>
<td>finare</td>
<td>finare</td>
<td>finare</td>
<td>finare</td>
<td>finα</td>
</tr>
</tbody>
</table>

Character Shortforms

As presented in section 2.2, there are a number of Melin symbols that represent both an individual character, and a short or frequently used word, for example “ö” and “och”. In the character shortform encoding scheme, both of these occurrences are mapped to the same symbol. During the decoding step, all occurrences of the respective character shortform, cf. Table 4.4, that are enclosed by blanks, i.e. individual characters, are replaced by the corresponding word. Symbol occurrences at any position within a word are retained as the corresponding single character. To exemplify this further, consider the encoded phrase “j jobbar”. The first “j” is isolated by blanks and is therefore replaced by the corresponding word “jag”, whereas the “j” in the beginning of “jobbar” is preserved as single character⁴. As indicated in section 2.2, other shortforms exist and some of them may be used as affixes. However, we focus on the selected character shortforms and isolated words here, in order to ensure a distinct decoding. In addition to this, this encoding approach removes the challenge of transcribing the same visual entity as two different representations, as the differentiation is performed afterwards, based on the symbol’s context.

Suffixes

The Melin system defines a number of symbols that represent typical Swedish suffixes. Out of these, we have selected two prominent ones, “-ing” and “-are”.

⁴This also applies to the other shortform characters “a”, “b” and “o”
and encode them individually. In addition to this we consider the definite article endings “-et” and “-en”. These two suffixes are not represented by a specific symbol of their own, like the former two, but by appending a “t”, respectively “n”, to the word, omitting the “e”, and thus also resulting in one-to-many mappings.

Table 4.4. List of short forms and the character by which they are replaced during the short form encoding step.

<table>
<thead>
<tr>
<th>av</th>
<th>bar</th>
<th>de</th>
<th>en</th>
<th>från</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>har</td>
<td>jag</td>
<td>kan</td>
<td>men</td>
<td>och</td>
<td>o</td>
</tr>
<tr>
<td>ut</td>
<td>var</td>
<td>är</td>
<td>över</td>
<td>ö</td>
<td></td>
</tr>
</tbody>
</table>

N-Grams
This encoding scheme replaces all occurrences of the n-grams, shown in Table 4.5, with individual symbols, as demonstrated in Table 4.3. Replacements are performed in a language-agnostic fashion, i.e. any combination of the respective characters (in the correct order) is transformed, even if the character combination would not be written with the respective n-gram symbol, according to Melin’s system. One example, where this naive approach may, on occasion, lead to incorrect replacements is compound words, where the n-gram is formed due to the merging of the two words, for example samma[n + s]ättning.

Table 4.5. List of n-grams considered for the n-gram encoding scheme. N-grams are resolved from biggest to smallest, to avoid conflicts due to overlaps, e.g. "nskt" vs "skt".

<table>
<thead>
<tr>
<th>nsion</th>
<th>nskt</th>
<th>nsj</th>
<th>nst</th>
<th>nsk</th>
<th>nkt</th>
<th>skt</th>
<th>ng</th>
</tr>
</thead>
<tbody>
<tr>
<td>sj</td>
<td>tj</td>
<td>br</td>
<td>fr</td>
<td>fr</td>
<td>kv</td>
<td>tv</td>
<td>sk</td>
</tr>
<tr>
<td>sl</td>
<td>sm</td>
<td>sn</td>
<td>sp</td>
<td>st</td>
<td>sv</td>
<td>nt</td>
<td>nd</td>
</tr>
<tr>
<td>ns</td>
<td>nj</td>
<td>nk</td>
<td>bt</td>
<td>pt</td>
<td>kt</td>
<td>xt</td>
<td></td>
</tr>
</tbody>
</table>

Melin
The final encoding scheme combines several aspects of the writing system and is thus termed Melin. Concretely, we consider a selection of shortforms, prefixes, suffixes and other n-gram symbols. The complete list is presented in Table 4.6.

4.3.2 Initial Encoding Results
In the initial set of experiments, each of the encoding schemes is applied individually during training. The size of the output symbol set is adjusted according to the number of symbols in the respective encoding, for example
Table 4.6. Summary of the Melin encoding scheme. Groups from top to bottom: short forms, prefixes, suffixes, n-grams.

<table>
<thead>
<tr>
<th>alldeles</th>
<th>bland</th>
<th>bättre</th>
<th>de</th>
<th>den</th>
<th>det</th>
</tr>
</thead>
<tbody>
<tr>
<td>då</td>
<td>där</td>
<td>efter</td>
<td>eller</td>
<td>en</td>
<td>ett</td>
</tr>
<tr>
<td>genom</td>
<td>gång</td>
<td>har</td>
<td>honom</td>
<td>här</td>
<td>ingen</td>
</tr>
<tr>
<td>inte</td>
<td>jag</td>
<td>just</td>
<td>kan</td>
<td>kom</td>
<td>kunna</td>
</tr>
<tr>
<td>med</td>
<td>men</td>
<td>mot</td>
<td>mycket</td>
<td>måste</td>
<td>och</td>
</tr>
<tr>
<td>om</td>
<td>själv</td>
<td>skulle</td>
<td>som</td>
<td>under</td>
<td>upp</td>
</tr>
<tr>
<td>var</td>
<td>varit</td>
<td>är</td>
<td>även</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| an- | be- | fort- | fram- | hän- | in- |
| kon- | kun- | kän- | lång- | märk- | någo- |
| på- | slut- | särskil- | till- | tänk- | ut- |
| verk- | vill- | över- |

-ande | -de | -are | -het | -kring | -ning |

-expanding it by 14 for character shortforms. Comparing the obtained results with the encoding-free baseline, it can be noted that the performance does not differ considerably. An exception to this is the WER obtained by the Melin encoding scheme, which yields a reduction of around 3 pp.

One concern in applying these encodings, using the original training lines of the LION dataset, is that several of the considered (sub-) words occur very infrequently, with a few tens of examples out of the almost 10,000 words. In order to investigate the potential of the encoding schemes further, we expand the training set with synthetically created text lines, described in more detail in the following section.

4.3.3 Expanded Dataset Results

In order to increase the raw number of samples for each word in the training set, we created a number of synthetic lines, based on the original data. For this, the original text lines were segmented, using the manual annotations, included in the dataset. The obtained images were combined with the line’s transliteration, under the condition that the number of segmented words in each modality matched. The 20 lines for which the number of segmented images and transliterated words did not match were discarded. Based on the annotated images, new lines were created by randomly shuffling the words and pasting them onto blank canvases. Line breaks were introduced such that

---

5Given the mostly well-separated handwriting of Lindgren, traditional segmentation approaches, for example via profile projection, should yield similarly usable segmentations.
the average character count roughly matched that of the original data. Overall, each original word image was used ten times in varying contexts to create new training lines.

The four encoding setups, as well as the encoding-free baseline were trained using the newly generated lines. Since the line generation did not take linguistic aspects into account and therefore created nonsensical texts, all models were fine-tuned on the original training set, afterwards. Overall, the combination of pre-training and fine-tuning yielded considerable improvements for all encodings (CER: 3.8 - 5.3 pp, WER: 6.4 - 8.3 pp), compared to their respective counterparts, trained only on the original data. Furthermore, the character shortform and Melin encoding schemes resulted in significant performance improvements, with respect to the encoding-free model.

4.3.4 Impact of Editorial Marks

As a final set of experiments, Paper IV briefly examines potential challenges of recognising struck and added lines. Using the encoding-free model weights from the previous experiment, the recognition performance for lines, bearing editorial marks, is obtained. In both cases, the CER and WER increase drastically, by 30 - 40 pp each. The challenges and potential solutions, surrounding the handling of struck-through words have been extensively discussed in chapter 3. Regarding lines with additions, it can be noted that the employed CTC-based architecture is not well-suited to handle these text arrangements. Concretely, the structure of the output assumes one character per timestep. Even if two characters could be identified, for example by taking the top-2, instead of top-1, symbols, the output lacks the spatial information, required to resolve which part belongs to the baseline text and which to the addition. This would be further complicated for cases where the baseline text has two (or more) additional layers of annotations. A straightforward approach to transliterating these additions is to segment the different layers before applying the text recognition model independently to each identified portion. Paragraph- or page-based approaches, such as the ones proposed by Bluche [9], and Yousef and Bishop [88], also present an interesting alternative but may require additional processing or reformatting of the existing dataset.

4.4 Conclusions and Perspectives

This chapter has presented the work performed in Papers III and IV. The respective conclusions can be summarised as follows:

- a variety of HTR and DIP augmentation techniques can be used in the context of handwritten stenography recognition without having an identifiable negative impact.
handwritten stenography can be recognised to some extent, using state-of-the-art HTR approaches

- the HSR performance can be improved via a combination of stenography-based encodings and a pre-training stage
- the presence of strikethrough and additions can negatively affect the recognition performance

All of the aforementioned conclusions are based on one single-writer dataset, which introduces limitations that need to be considered. Working with multi-writer datasets can generally be expected to increase the complexity of the recognition task, due to natural variations in handwriting style. For handwritten stenography, this multi-writer effect is further exacerbated by the personal style with respect to the choice of symbols and the development of custom signs. Unfortunately, it was not feasible to collect stenographic samples from other writers during the work on Papers III and IV, in order to perform an analysis of such variations. Despite these limitations, Papers III and IV, and especially the LION dataset, will hopefully encourage future research in stenography recognition, which may eventually lead to the collection of a more diverse dataset and the implementation of further methods.

Considering the presented works and the LION dataset themselves, several avenues for future work arise. Besides the exploration of other encoding schemes, the collection and use of true diplomatic transcriptions is of interest. The latter will require a large annotation effort, as well as the development of a precise annotation scheme, and the training of annotators. It is therefore likely not feasible to implement in its entirety. However, partial diplomatic transcriptions, for example for the most frequent or stenographically interesting words, should be easier to obtain and could serve as a first case study.

In addition to efforts closer to the data, it would be interesting to introduce linguistic aspects into the recognition. This can for example be in the form of a language model for the CTC decoding or the exploration of the phonetic components of stenography.

Considering the viewpoint of cultural heritage and its preservation, an extension of the experiments of other writing systems, and languages, is of special interest. An example for a very challenging but also interesting stenography project is the Dickens Code\textsuperscript{6}, which is concerned with the decoding of Charles Dicken’s shorthand notes, written in Brachygraphy.

As a final topic for future work, I would like to mention a work in progress, concerned with the reimplementation and reproduction of the paper by Zhai et al. [89]. A summary of the preliminary findings is attached in Appendix A. In my opinion, investigating the reproducibility of a work is not only a great way to take a closer look at certain details of the method, but it can also contribute to open science, by providing a public implementation where there may be

\textsuperscript{6}https://dickenscode.org/
none. Furthermore, it can contribute to building a culture of code sharing and keeping reproducibility aspects in mind.
5. Conclusions and Future Directions

This dissertation summarises work performed in computerised image processing, with a special focus on document images. The first set of papers is centred around the handling, and especially cleaning, of struck-through words. In the second set of papers, the focus lies on handwritten stenography recognition. Across all of these works, open, i.e. publicly available, data has been a central aspect. Furthermore, efforts have been made to promote reproducibility by making all related code available alongside the respective paper.

5.1 Summary of Contributions

The contributions of the four papers, that this thesis is based on, can be summarised as follows:

Paper I
- introduction of a novel strikethrough generation method
- presentation of two new, publicly available datasets for strikethrough removal – a synthetic one, created using the developed strikethrough generation method, and one consisting of genuine strikethrough
- identification of struck-through words and classification into different types, using state-of-the-art deep neural networks
- study of strikethrough removal, using unpaired image-to-image translation in the form of regular and attribute-guided CycleGANs [52, 91]

Paper II
- presentation of a new, publicly available dataset for strikethrough removal, based on a combination of the genuine dataset and strikethrough generation method, introduced in Paper I
- study of strikethrough removal, using paired image-to-image translation

Paper III
- study of different augmentation techniques, typically used in the context of HTR, for handwritten stenography recognition
Paper IV

• presentation of a novel, publicly available dataset for handwritten stenography recognition
• introduction of a baseline for handwritten stenography recognition, using a state-of-the-art text recognition deep neural network [21]
• integration of stenographic domain knowledge into the recognition process
• further improvement of results via a pre-training and fine-tuning scheme

5.2 Future Directions

A variety of potential topics for future work have been discussed in detail in the two previous chapters. To recapitulate, future directions with respect to strikethrough processing are largely founded in the creation and exploration of new datasets, for example containing genuine samples from multiple writers. Regarding handwritten stenography recognition, the expansion of the existing LION dataset and the investigation of other stenography systems are of interest. Other avenues for future work in this context are the collection of true diplomatic transcriptions and the integration of linguistic knowledge.

Considering the context of the Astrid Lindgren Code project, the combination of the two topics – strikethrough removal and handwritten stenography recognition – is of relevance.

Lastly, entirely independent of the aforementioned research areas, document (and other) image processing can only benefit from making research code and related datasets publicly available and investing into the integration of reproducibility aspects into the ecosystem. This can, for example, be in the form of an artefact evaluation step, which is already in use in other computational research areas.


Den här avhandlingen, som en del i projektet Astrid Lindgren-koden, presenterar metoder för automatisk transkribering av stenografi. I projektet har delar av Astrid Lindgrens manuskript transkriberats med hjälp av "crowdsourcing" och frivilliga stenografer. Med detta träningsmaterial som grund har metoder utvecklats för automatisk transkribering, där maskinen försöker imitera människan genom s.k. maskininlärning. Eftersom stora delar av Lindgrens efterlämnade material är anteckningar, har metoder för att identifiera överstrykningar varit centrala. Dessa ger en inblick i författarens tidiga tankar kring karaktärer och händelser, innan den utgivna versionen färdigställts. Den här sammanläggningsavhandlingen innehåller fyra vetenskapliga artiklar enligt följande:


1https://www.barnboksinstitutet.se/forskning/astrid-lindgren-koden/
I artikel II, som fortsätter arbetet från artikel I, presenteras en annan metod för att hantera överstrukna ord. Dessutom utvecklades och publicerades ett tredje dataset.

Artikel III undersöker användningen av så kallad ”data augmentation” för automatisk igenkänning av Lindgrens stenografiska manus. Tekniken ”data augmentation” går ut på att artificiellt utöka ett träningsmaterial med hjälp av att slumpmässigt modifiera orijinalmaterialet med realistiska synthetiska förändringar.

Slutligen presenterar artikel IV resultat rörande transkribering av stenografi. Den vanliga textigenkänningsprocessen utökas till att omfatta utvalda aspekter från stenografi, till exempel användningen av korta symboler för att representera hela ord. I samband med detta arbete presenteras också ett nytt dataset, som kallas LION (lejon), som innehåller utdrag ur Lindgrens stenografiska manus, bland andra från barnboken Bröderna Lejonhjärta (som inspirerade datasets namnet).
After five and a half years full of ups and downs, learning and teaching, and many a cup of coffee, this PhD journey now comes to an end. Along the way I met many great people who left a mark in one way or another and who shall be mentioned here.

Firstly, I would like to thank my supervisor, Anders Hast, for this PhD opportunity, interesting discussions, and food for thought about research and academia.

To my co-supervisor, Ekta Vats: Thank you so much for your good advice and feedback, and support and encouragement along the way!

Anders Brun, although your time as co-supervisor was unfortunately cut short, your input has been appreciated!

Fredrik Wahlberg, thank you for stepping up as co-supervisor, for your support, and for providing a different perspective!

Besides my supervisory team, many other people have provided input and support in various forms over the years. I am very grateful to Ida-Maria Sintorn, and my senior team, Carolina Wahlby and Åsa Cajander, for checking in on me and providing encouragement and support!

Much of this PhD and dissertation is connected to the Astrid Lindgren Code project. My deepest gratitude to Malin Nauwerck, for welcoming me to the team, many enlightening discussions about stenography, and patience and encouragement along the way, especially during the joint work on Paper IV! A big thank you also to Karolina Andersdotter, for good advice and advocating for open science!

Davide Vega D’Aurelio, thank you for being a great unofficial pedagogic mentor, inspiring teaching role model, and for spontaneous coffee and office chats, not just about work!

To Lars Oestreicher and Kiko Fernandez-Reyes, thank you for your patience, while I was finding my bearings as TA in your courses, and to Mikael Laaksoharju for getting me in touch with the right people, so that I could teach the courses that I was interested in!

Thanks also to Robin Strand and Ingela Nyström for their help with wrapping things up, and to everyone else at Vi3.

This journey was made so much better by all the brilliant PhD students, PostDocs and visitors I got to meet!

A big thank you to Nicolas Pielawski, a great discussion partner for topics far and wide, skilled mushroom hunter and fun office mate!
Massive thanks to Nikolaus Huber, for the company and support during weekends and the occasional late night at university, for rants about life, but also for movie nights, walks and many delicious baked goods!

To Eva Breznik, thank you so much breakfasts and chats, for encouragement and for picking me back up when things got tough!

Many thanks to Tobias Mages, for encouragement, especially during the last weeks of dissertation writing, and to Peter Münch, for Sunday lunches at university and many interesting chats! Elisabeth Wetzer, thank you for fun times working on the SSBAktuell newsletter and the great company at conferences! To Nadezhda Koriakina, for many fun coffee and lunch breaks, for knitting meet-ups and encouragement! Virginia Grande Castro thank you for your kind words, lunch visits and a great time in the thesis writing group! To Johan Snider and Bedour Alshaigy for fun chats and random walks around the university! And to Bozhi Liu for kind words and a fun mushroom-picking excursion!

Many thanks to Nikolaus Huber, Tobias Mages and Elli Anastasiadi for cinema and nation visits, and making sure there is a life outside of work.

A big thank you to all the PhD students that came before me, especially Håkan Wieslander, Teo Asplund, Mariëlle Jansen, Fredrik Nysjö, Leslie Solorzano, Johan Öfverstedt, Tomas Wilkinson and Kalyan Ram Ayyalasomayajula, and to those that joined later, especially Ankit Gupta, Axel Andersson and the new SSBAktuell editors, Can Deniz Bezek and Andrea Behanova.

I would also like to thank all collaborators from the humanities, especially Lasse Mårtensson, Anna Fredriksson and Jacob Orrze, for taking the time to answer my questions, and allowing me to catch a glimpse of their work.

Many thanks also to the people behind the scenes, Astrid Raidl, Ulrika Jarensund, Marina Nordholm, Anna-Lena Forsberg, Loreto Skoknic and everyone else in administration and IT.

To what is left of the syjunta group, Li Hedenmalm and Saga Samuelsson, thank you for arts and crafts, fair excursions, and coffee chats!

To the Vi3 kids, Émile and Léon, Sever, Efra and Viktoria, for smiley faces and the occasional playground visit, and especially for giving their parents a break every now and then, so that they can have a coffee with me.

Through a series of fortunate coincidences, I was lucky to find a teaching community outside the regular university course environment but still within academia, leading me to meet many inspiring people. A big thank you to the Carpentries and Code Refinery communities, and especially Annika Rockenberger, Thor Wikfeldt, Paula Martinez Lavanchy and colleagues, and of course my co-instructors from the DHNB 2022 workshop, Diana Iusan, Radovan Bast, Henrik Askjer and Kristoffer L. Nielbo. It has been a great pleasure!

Many thanks also go out to my new colleagues at Folkrörelsearkivet för Uppsala län and Arbetarrörelsens arkiv och bibliotek for a kind welcome to
the team and encouragement over the last months.

A massive “thank you” to all of the stenography volunteers, who transliterated and proofread Astrid Lindgren’s manuscripts, and without whom large parts of this dissertation would not have been possible!

Last, but definitely not least, deepest gratitude to my parents, Reynaldo Heil and Elisabeth Villaret-Heil, and my sister Franziska Heil, for their encouragement and being understanding when work took precedence over spending time with family. Many thanks also to all of my friends back home, Frank, Sabine, Julia, Christin, Katharina, Lisa, Nina, Jasmina and everyone else, for reminding me to take a break sometimes.

Resources

I gratefully acknowledge the PhD funding, provided by Uppsala University and especially the Vi3 division. A small portion of this PhD employment was also funded by the project *Marginalia and Machine Learning* (PI: Peter Heslin, Durham University).

This work is partially supported by Riksbankens Jubileumsfond (RJ) (Dnr P19-0103:1).

The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Chalmers Centre for Computational Science and Engineering (C3SE), the Umeå High Performance Computing Center North (HPC2N), and the Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.


[19] François Chollet et al. Keras. https://keras.io, 2015.


2021. Springer International Publishing.


Skrifter utgivna av Svenska Barnboksinstitutet

Studies published by the Swedish Institute for Children’s Books
ISSN 0347-5387
Tidigare titlar, se www.barnboksinstitutet.se


Appendix A. Exploring Reimplementation and Reproducibility of the Optical Gregg Shorthand Recognition

In 2018, Zhai et al. presented a work on stenography recognition, based on word images, digitised from a dictionary for Gregg’s shorthand [89]. They published their dataset and a portion of their code in conjunction with their paper. Following the initiative, put forward by the ReScience C journal\(^2\), I have been working on a replication of the work by Zhai et al. [89]. Although this work is still in progress, a partial replication has been achieved, which will be outlined below.

I include this work in progress in this dissertation in the hope that it will promote code sharing. Furthermore, I hope to encourage others to share their replications, as many researchers in our field have, at one point or another, had to reimplement an algorithm or network, but may have never taken steps to share this important work.

In Brief: Original Method

Zhai et al. propose a neural network that transcribes the content of a word image, character by character, and combine the obtained word hypothesis with different retrieval schemes. The authors do not employ a classical CTC setup but instead use a time-step-wise cross entropy loss. Furthermore, they consider both a forward hypothesis, i.e. left-right reading order, and a backward hypothesis in the opposite direction. [89]

Scope of the Replication

I began my replication by identifying the test split of the original work. The repository\(^3\), shared by Zhai et al. contains two sets of word indices, and corresponding recognition results. One of these was confirmed to be the test set by evaluating it via the provided evaluation scripts. Based on the identified split,

\(^2\)http://rescience.github.io/
\(^3\)https://github.com/anonimously/Gregg1916-Recognition
a replication of the presented metrics was attempted. Afterwards, the neural network architecture was reimplemented in PyTorch [67], as compared to the original implementation in Keras [19].

Reimplementation of Evaluation Metrics

Using the identified test set and the included recognition results as unit tests, the results of the first two rows in Table 2 of [89] were studied.

A reimplementation of the accuracy was straightforward and the results in the table could be reproduced without issue. Regarding the editorial similarity ($esim$), it has to be noted that the original implementation does not fully correspond to the definition provided in the paper. Concretely, the authors state:

$$esim(ref, hyp) = 1 - \frac{edist(ref, hyp)}{\max(|ref|, |hyp|)}$$  \hspace{1cm} (5.1)

where $edist$ is the Levenshtein distance [50], $ref$ the reference (ground truth) string, $hyp$ the hypothesis, produced by the model, and $|.|$ the length of the respective string. However, the implementation that they use, and that produces the results, shown in the Table 2, corresponds to:

$$esim(ref, hyp) = \max(0, 1 - \frac{edist(ref, hyp)}{|ref|})$$  \hspace{1cm} (5.2)

Using the latter definition, extracted from the code, the $esim$ results for the first two rows of Table 2 can be reproduced.

Regarding the BLEU-1 to BLEU-4 score [65], it should be noted that the authors use the implementation provided by NLTK [8], in combination with NIST geometric sequence smoothing (“smoothing method 3”) [17]. This information was not provided in the paper but again could be extracted from the companion code without issue. Using the aforementioned implementation, the respective portion of rows 1 and 2 could also be reproduced, with a minor discrepancy – possibly due to a manual data entry error. The reported values for BLEU-3 and 4 for the forward hypothesis differ by 0.002, each, compared to the reproduction.

Reimplementation and Partial Evaluation of the Proposed Architecture

Based on the information provided in the text and in Figure 3 of [89], the proposed architecture was reimplemented. The model was trained, following the protocol, outlined in the paper, using the exact augmentation protocol, as given in the code.
At the time of writing, only the raw forward hypothesis (row 1 in Table 2), has been examined. Concretely, the model was trained 50 separate times and evaluated via accuracy@1 (acc@1) and editorial similarity ($e_{sim}$). The obtained results, as well as the originally reported values, are shown in Table 5.1. In the case of accuracy@1, the mean performance matches the original results almost perfectly. Regarding the editorial similarity, a small difference can be observed, however the results are still in the same general range.

In order to place the performance of this recognition model in relation to HTR in general, and to the work presented in chapter 4, the CER was also computed for the replication experiment. On average, the architecture achieved a CER of 46.33% (SD = 0.2).

Table 5.1. Preliminary reproduction results for the forward hypothesis.

<table>
<thead>
<tr>
<th>Metric</th>
<th>$e_{sim}$ (SD)</th>
<th>acc@1 (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.574 (N/A)</td>
<td>0.027 (N/A)</td>
</tr>
<tr>
<td>Ours</td>
<td>0.539 (0.013)</td>
<td>0.028 (0.006)</td>
</tr>
</tbody>
</table>

Preliminary Conclusions and Future Work

Based on the results, obtained so far, I believe a case can be made that the work has, at least in part, been successfully replicated. Although some of the details had to be extracted from the companion code, the most important pieces of information were provided in the paper.

Regarding the remaining replication work, it has to be noted that the implementation and investigation of the backward hypothesis is currently in progress. As this portion of the recognition is required for the retrieval-based experiments, the latter work is currently on hold.
Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2294

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title “Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology”.)