Verwendung von Synthetischen Datensätzen für Anwendungen im Bereich der Dokumentenanalyse

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Markus Muth, BSc.
Matrikelnummer 01127793

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Dipl.-Ing. Dr.techn. Florian Kleber
Univ.Ass. Dipl.-Ing. Marco Peer, BSc

Wien, 27. September 2023

Markus Muth
Robert Sablatnig

Informatics

Synthetic Data for Applications in Document Analysis

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Markus Muth, BSc.
Registration Number 01127793

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: Dipl.-Ing. Dr.techn. Florian Kleber
Univ.Ass. Dipl.-Ing. Marco Peer, BSc

Vienna, 27th September, 2023

Markus Muth

Robert Sablatnig

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Markus Muth, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. September 2023

Markus Muth

Danksagung

Ich möchte mich bei meiner Familie, Freunden und meiner Frau bedanken, die mich im Rahmen des Schreibens dieser Masterarbeit mit ihren Hilfsangeboten und aufmunternden Worten unterstützt, und meine Launen geduldig ertragen haben. Ein Kompliment meinen Betreuern, die mich mit ihrer Geduld und ihrem Wissen geleitet haben.

Acknowledgements

I want to thank my family, friends, and wife, who supported me in all aspects during my journey of writing this thesis – thank you for your help, supporting words and for bearing my moods. I want to compliment my supervisors, who guided me with their patience and knowledge.

Kurzfassung

Die Verwendung von synthetischer Handschrift zur Verbesserung von Machine Learning Methoden wird für zwei Anwendungsbereiche analysiert: das Finden von Handschrift in Bildern (Handwritten Text Detection, HTD), und die Zeichenerkennung für Handschrift (Handwritten Text Recognition, HTR). Für HTD wird synthetische Handschrift mithilfe eines bereits vorhandenen Machine Learning Modells [DMP+20] generiert und zu gescannten Dokumenten hinzugefügt, um handschriftliche Notizen zu imitieren. Modelle zur Objekterkennung (YOLOv5 [JAS+] und YOLOv8 [JCQ]) werden trainiert, um die Handschrift vom restlichen Inhalt der Dokumente zu unterscheiden. Anschließend werden diese Modelle mit echten Daten evaluaiert: Für den CVL Datensatz [KFDS13] wird eine mAP@50 von 0.88 und ein F1@50 auf Pixelebene von 0.96 erreicht; für echte Notizen auf einer wissenschaftlichen Publikation wird eine mAP@50 von 0.72 und ein F1@50 von 0.89 auf Pixelebene erlangt. Die synthetisch generierte Handschrift wird weiters verwendet, um ein bereits vorhandenes Modell zur Zeichenerkennung [CCP21] zu trainieren. Anschließend wird dieses Modell zur Erkennung des Inhalts von Bildern mit echter Handschrift angewendet. Dies führt zu einer Zeichenfehlerhäufigkeit (Character Error Rate, CER) von 28.3% und einer Wortfehlerhäufigkeit (Word Error Rate, WER) von 65.5% für Bilder aus dem IAM-Datensatz [MB02], was mehr als dreimal so hoch ist wie die Fehlerraten ohne synthetischen Daten. Die Verwendung von synthetischen Bildern in Kombination mit echten Daten ermöglicht jedoch eine Reduzierung der Fehlerraten im Vergleich zur Verwendung von echten Daten allein, insbesondere für kleine Datensätze. Die Verwendung von nur 10% der Trainingsdaten (113 Bilder) aus dem CVL-Datensatz [KFDS13] führt zu einer CER von 54.5% und einer WER von 88.8%. Wenn das Modell jedoch mit synthetischen Daten vortrainiert wird, ergibt sich eine CER von 14.6% und eine WER von 43.4%.

Abstract

The usability of synthetic HandWritten Text (HWT) to improve machine learning models is assessed for two domains: Handwritten Text Detection (HTD) and Handwritten Text Recognition (HTR). Synthetic HWT is generated using an existing model [DMP$^+$20], and added to scanned documents to mimic handwritten annotations. Object detection models (YOLOv5 [JAS$^+$] and YOLOv8 [JCQ]) are trained to distinguish HWT from remaining content. Applying those models to real data results in a mAP@50 of 0.88 and a pixel-level F1@50 of 0.96 for the CVL data set [KFDS13], and a mAP@50 of 0.72 and F1@50 of 0.89 for a scientific paper with real handwritten annotations. The synthetic HWT from [DMP$^+$20] is further used to train the HTR model described in [CCP21], which is then applied to recognize the content of real HWT data sets. This results in a Character Error Rate (CER) of 28.3% and a Word Error Rate (WER) of 65.5% for line images of the IAM data set [MB02], which is more than three times higher than the state-of-the-art results. However, mixing synthetic with real data allows to reduce the CER and WER compared to using real data only, especially for small data sets. Using only 10% of the training data (113 images) from the CVL data set [KFDS13] results in a CER of 54.5% and a WER of 88.8%, pre-training the model with synthetic data results in a CER of 14.6% and a WER of 43.4%.

Contents

CHAPTER 1

Introduction

HandWritten Text (HWT) is an important part of many societies, be it in the form of gift cards, letters, or study notes. Processing it is crucial for applications like signature verification [SLY22], searching in handwritten documents [KDJ16], or transcribing HWT into a machine-readable format [CCP22].

HWT, especially compared to Computer Written Text (CWT), shows a wide variety of styles. They not only differ between persons but also within the same individual, e.g., due to different pens or distractions during writing [FS15]. HWT itself can be described on various properties, such as the writing zones (size of lower-case and upper-case characters, and lower-case characters that go above/below the baseline of HWT), the slant of characters, or the width of different characters, especially white spaces [MMB01]; using different pens can affect the stroke width and stroke color. Spacing between lines, line rotations, the slant of the base line, or line indentations are possible characteristics to describe paragraphs of HWT.

HWT must be digitized to be usable in a computer-aided fashion. This can be done in two ways: online HWT, and offline HWT. Online HWT is usually obtained by tracking and sampling the movements of a stylus or similar means of input on electronic devices such as tablets or whiteboards (the latter was used for the IAM Online Handwriting Database [LB05]), resulting in a sequence of Cartesian coordinates. To interpret the data, the tracked points can be connected via, e.g., Bézier curves to form the stroke of HWT. Offline HWT is acquired by creating photographs or scans of HWT.

One method for transcribing offline handwritten text is to locate areas containing such within the image, segment it into individual lines of text, and finally into individual words. Those can then be recognized and put together to recreate the text [KGS99]. Deep learning methods allow for reducing the required granularity for segmentation by transcribing images of entire lines or entire paragraphs [CCP22] in one step.

However, developing deep learning models usually requires a lot of training data (e.g., Fogel et al. [FAEC$^+$20] used a data set of more than 100k images for a word recognition task and achieved a word error rate of 29.75% on the IAM Handwriting Database [MB02]). In the mentioned domain of HTR this means that images containing HWT have to be labeled, possibly involving the collection of specimens of handwriting. This can be a time-consuming process requiring a significant number of people – domain experts, for example, which are trained in reading historic documents (as needed for transcription of such documents, e.g., [SRTV16]), or volunteers providing specimens of their handwriting (as needed for HWT data sets like the IAM database [MB02] where more than 600 writers contributed).

Advances in deep learning models allow to reproduce HWT synthetically. Graves [Gra13] developed a model based on the Long Short-Term Memory (LSTM) architecture to predict the Cartesian coordinates of online HWT for different input text and styles. Brian et al. [DMP$^+$20] developed a Generative Adversarial Network (GAN) to generate entire images containing HWT for different styles and arbitrary text.

The usage of synthetic data can be beneficial for HWT processing research since it allows to reduce the efforts needed to obtain high-quality training data: the ground truth of the synthetic data, whether the locations of synthetic HWT for HTD or the actual transcription for HTR, is implicitly available. It is also possible to generate a pixel-accurate ground truth for HTD. Synthetic data allows to generate arbitrarily large data sets, as no human interaction is needed for labeling the data. Further, recent research has shown that using computer-generated images of HWT can help to improve the quality of word-level transcriptions [FAEC$^+$20] (the word error rate was improved from 12.24% to 11.68% for the IAM Handwriting Database [MB02], and from 24.73% to 23.98% for the RIMES data set [GCBG09]), or to segment HWT vs. CWT on pixel-level [JKSC20].

Therefore, this thesis aims to further evaluate to which extent synthetic HWT can be used to improve machine learning models working with HWT data. The domain is narrowed down to two tasks. The first one is Handwritten Text Detection (HTD), which aims at identifying areas of handwritten text in documents. Developing good models in this area can have advantages for, e.g., form parsing (especially for HWT written outside of the predefined regions) or for translating notes in various documents – i.e., documents containing hand-written annotations – into a machine-readable format. The second task is Handwritten Text Recognition (HTR), which means translating the content of an HWT image into a machine-readable sequence of characters. Applying synthetic data for model training could improve the overall model performance or allow a better writer-specific HTR.

More precisely, the following research questions are to be answered:

RQ1 To which extent can synthetic handwritten text be used to improve handwritten text detection models for annotated documents?

RQ2 To which extent can synthetic handwritten text be used to improve handwritten text recognition models on line- and paragraph-level?

On a high level, the methodology for answering the questions is the same for both: First, various task-specific data sets containing synthetic HWT text are created. The synthetic images for those data sets are generated with existing HWT synthesis models. Then, existing machine learning models suitable to solve the respective tasks are trained on the generated data. The suitability of synthetic data is evaluated by assessing the model performance using real data.

The main contributions of this thesis are the following:

- A method for generating synthetically and realistically annotated documents is selected and evaluated
- Using synthetic data for HTD in annotated documents in combination with state-of-the-art object detection models is evaluated
- Using synthetic data for HTR on line- and paragraph-level using state-of-the-art HTR models is evaluated

The remaining of this work is organized as follows: Chapter 2 gives an overview of related work. Section 2.1 describes different methods of generating HWT, including the approach used within this thesis. Section 2.2 and Section 2.3 describe the existing models for HTR and HTD, which will be used for evaluating the applicability of synthetic data sets. Section 2.4 gives an overview of existing approaches to use synthetic HWT to improve machine learning models. Chapter 3 describes in detail how the synthetic data sets are generated; Section 3.1 describes how the synthetic data used within this thesis is created, Section 3.2 covers the generation of synthetically annotated documents, and Section 3.3 focuses on the generation of synthetic data for HTR. Chapter 4 describes the evaluation results for HTD (Section 4.1) and HTR (Section 4.2). A summary of the key insights can be found in Chapter 5.

CHAPTER 2

Related Work

The topic of this thesis overlaps with four different research areas, which are HWT generation, HTD, HTR, and using synthetic HWT data sets to improve models. Section 2.1 gives a brief overview of the history for different approaches to generate HWT and describes the model architecture that is used within this thesis. Section 2.2 describes the HTR model utilized for the evaluation workflow. Section 2.3 briefly describes the object detection architectures for HTD; Section 2.4 outlines existing work regarding the usage of HWT for HTD and HTR tasks.

2.1 Handwritten Text Generation

Various approaches exist for generating synthetic HWT, which either generate synthetic online HWT text or entire images mimicking offline HWT data. Graves [Gra13] developed a method mixing a long short-term memory model with Gaussian mixture models utilizing online HWT data. The model predicts a series of three values based on the last prediction. The first two values are Cartesian coordinates (actually the offset to the previous predicted point), which, after connecting the coordinates, can be interpreted as HWT. The third value indicates the end-of-stroke, i.e., whether the pen was lifted. Without further constraints, the model will predict a sequence of points that look like handwriting but do not necessarily form valid words. Dynamically weighting the characters of the input character sequence ensures that the generated point sequence represents the text on the input sequence. Restricting the standard deviations of the Gaussian mixture models increases the smoothness of the generated strokes, and initializing the model with real online HWT enables the model to predict text coordinates mimicking the given style.

Since this approach generates a sequence of points, the actual stroke must be generated by connecting the points, e.g. using Bézier Curves. Varying the color and thickness of the curve allows the generation of HWT with different stroke colors and widths. However, varying pen pressure, hence varying stroke intensities and widths, can not be mimicked with this approach.

This drawback can be solved by generating entire images containing HWT. Kang et al. [KRW+20] developed one such solution utilizing GANs consisting of two encoders and one generator to generate images of arbitrary (out of a fixed alphabet) but length-limited text. The first encoder extracts style features from real HWT images. The second one learns textual features, allowing the generation of images of words not in the training data. The generator combines information from both encoders to generate images with a given style and content. The model is optimized in three ways. The first one is the for GANs typical discriminative loss aiming to distinguish real HWT images from generated ones. A style classifier ensures that the generated images have a wide variety of different styles. A HTR model is utilized to ensure that the content of the images match the input string.

The limitation of input strings with a maximum length is solved with the architecture proposed in [FAEC+20]. An important assumption in this work is that creating HWT is a local process, i.e., a character is influenced only by its direct neighbors. As before, the network consists of three parts: a generator, a discriminator, and a recognizer network. However, instead of generating the entire image at once, the generator focuses on patches of 16x32 pixels [1], which are concatenated for arbitrary-length text input. The receptive fields of the generators overlap to account for smooth transitions between letters, especially with cursive HWT. Each patch is generated by identical, but on the respective character conditioned, generator. The discriminator is again operating on smaller patches with overlapping receptive fields to learn to distinguish real from generated HWT. The recognizer is an LSTM pre-trained on real HWT images without learning an implicit language model to promote the training of realistic HWT images.

Using an explicit space predictor network, which takes a multi-dimensional style vector and a character string as input and predicts the character widths in the generated image, allows Davis et al. [DMP+20] to generate HWT images with arbitrary length and varying character widths. The entire network architecture is again composed of multiple smaller networks, additionally to the space predictor. A fully convolutional recognition network is pre-trained on real images for character predictions. The output of the recognizer network, together with the actual image, is fed into a style extractor network, which generates a vector containing global and character-level style features. The spacing information (which includes the text to be converted to HWT), together with the style vector, are the input for the generator network, which finally outputs the synthetic HWT image. The pre-trained recognizer network and a discriminator network trained to predict real vs. generated images are appended to the generator to lead the generator towards realistic-looking images and correct text content.

Bhunia et al. [BKC+21] followed an approach similar to [DMP+20] but with essential differences in the style encoding. First, instead of locating the characters within the style

[1] The width of the patches is 16 pixels, the height is 32 pixels.

feature sequence to apply character-specific style extraction, the individual vectors of the feature space cover a more generic region of the input image. Secondly, while [DMP+20] employs a convolutional network on image batches as a discriminator network, a cycle loss [ZPIE17] is applied by Bhunia et al. to ensure that the generated image allows the reconstruction of the encoded style features. Both enable the model to learn more fine-grained local peculiarities of different HWT styles, such as ligatures.

The models proposed in [FAEC+20], [DMP+20] and [BKC+21] are all trained, among others, on the IAM Handwriting Database [MB02], which contains handwriting samples of 657 writers and, in total, 9682 text lines split up into training, validation, and test images. [FAEC+20] achieves a Fréchet Inception Distance (FID) and a Geometry Score (GS)[2] of 23.78 and 0.00076, respectively; The model proposed in [DMP+20] achieves a FID of 20.65 but a GS of 0.0488. An evaluation with humans is additionally reported for [DMP+20]: real and generated images were presented to humans, who had to decide on the image's origin (real vs. synthetic). Considering real images only, 34.2% are correctly recognized as human origin, 15.8% are wrongly identified as computer-generated. Considering synthetic images, 31.9% are wrongly identified to be of human origin (which is good, as this means a realistic appearance of the text), and 18.0% of the images are correctly identified as synthetic[3]. [BKC+21] achieves a FID of 19.40 and a GS of 0.0101; a human study was executed as well: 24.9% of real images are correctly identified as such, 25.1% are predicted to be of synthetic origin. 26.8% of generated images are predicted to be real, and 23.2% are correctly identified as synthetic HWT.

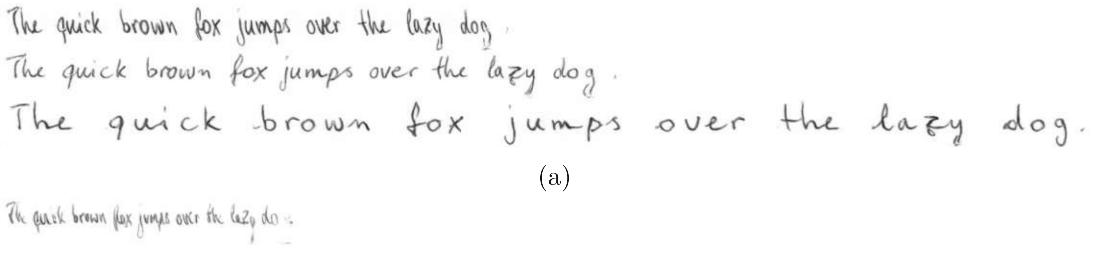
The model proposed by [DMP+20] is used throughout this thesis for generating synthetic images due to its ability to handle HWT content of arbitrary length and to generate data with varying character widths. [BKC+21] satisfies both conditions as well. However, the qualitative studies reveal that, while humans can not distinguish generated from real images better than random guessing (52.2% correct predictions for [DMP+20] vs. 48.1% for [BKC+21]), synthetic images from [DMP+20] are more often guessed to be of human origin than for [BKC+21] (31.9% vs. 26.8%). Figure 2.1a shows some synthetic example images generated with the model from [DMP+20].

2.2 Handwritten Text Recognition

Early attempts for HTR systems were based on identifying single characters within an image, recognizing them, and concatenating the results to the final transcriptions [KGS99]. Advances in deep learning allow the recognition of words [CMV+19], lines, and entire paragraphs [CCP21]. Some of such models [KGS99] rely on an explicit segmentation on word- or line-level during training, which requires the preparation of proper labels. However, recent approaches [CCP21] allow the training on entire paragraphs, without any visual segmentation into lines, words, or characters; only the recognized text is expected to be split accordingly into individual lines.

[2] Lower values are better for both metrics.

[3] The numbers add up to 99.9%, as it does in the original paper.


The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy do

(a)

(b)

Figure 2.1: Synthetic HWT images generated by the model developed by Brian et al. [DMP+20] using the input string “The quick brown fox jumps over the lazy dog.” and four different styles. Notable are varying appearances of the text in (a) (e.g., the letter “T” is written differently, the character slants or character widths are different). (b) shows an image with a quite narrow font – the smaller the character widths, the less legible the text tends to become.

For this thesis, the state-of-the-art segmentation-free HTR model for paragraphs proposed by Coquenet et al. [CCP21] is used. Segmentation-free means no explicit labels for characters, words, or lines are needed, as the model learns them implicitly during the training process. This is achieved using a Vertical Attention Network (VAN) architecture, which recurrently identifies individual lines of a paragraph via self-supervised implicit line segmentation. The architecture consists of three main parts. The first one is a fully-connected network acting as an encoder, which extracts features from the input image preserving the two-dimensional structure of the data.

The next block is an attention module, which operates on the encoded features to (1) identify single lines in their correct order, and (2) detect the end of the paragraph. The former happens not only on the encoded feature space but also on the history of previous attentions and recognized content. Three different strategies are proposed for the latter. First is the “fixed-stop” approach, where a fixed number of line features are considered. The decoder module, responsible for recognizing the text content, predicts an empty string in case no text is recognized in a line feature set. The second approach, called “early-stop”, is to stop once the first empty line is recognized. The third method, called “learned-stop”, is to learn when to stop by predicting the probability wheter a specific line should be considered for recognition.

The third module of the network is the decoder, which operates on line features, i.e., the subset of the encoded features selected by the attention module. The decoder consists of an LSTM and a convolutional layer for the final character predictions.

Parts of the VAN, more precisely the encoder and the convolutional layer of the decoder, can be pre-trained on line images to speed up the training process on paragraph images. With this approach, a character error rate of 4.45% and a word error rate of 14.55% are achieved on the IAM Handwriting Database [MB02]. This method supports cross-data set training, where the line-level images for pre-training are from a different data set than the paragraph images. Although this approach does not yield a better character error rate, the final results are not more than 1% worse than with inter-data set training.

The model's ability to achieve state-of-the-art results on line- and paragraph HWT recognition (a character error rate of 4.45% and a word error rate of 14.55% are achieved on paragraph images of the IAM Handwriting Database [MB02]; a more detailed comparison of different approaches can be found in [Coq22]), the possibility to work on line- and paragraph data separately, and the freely available source code are the reasons why the model proposed in [CCP21] is chosen as HTR model for this thesis.

2.3 Handwritten Text Detection

Jo et al. [JKSC20] proposed a convolutional neural network to segment HWT from CWT on pixel-level. The model is trained on synthetic data, where images of real HWT are added to scanned documents. Evaluation is done on real data, where Optical Character Recognition (OCR) is applied on the documents after the detected HWT being removed, which results in an accuracy (ratio of correct characters over correct + incorrect + missing characters) of 92.5%. However, the text was added randomly across the entire page, whereas this thesis focuses on generating synthetic images with non-overlapping HWT and CWT. Related work was also done by Zagoris et al. [ZPA$^+$12], who applied the bag of visual words paradigm to identify handwritten text in scanned index cards and in images from the IAM Handwriting Database [MB02] and achieved a F1 score – normalized by an approximation of the number of characters – of 0.77 and 0.99, respectively.

A different approach is evaluated in this thesis: object detection models are trained to separate HWT from the remaining content of a scanned document. One-stage networks based on the YOLO [RDGF16] architecture are used for this purpose due to their state-of-the-art performance on the MS COCO data set [LMB$^+$14] (the AP@0.5 is 64.9, see [AYX$^+$23] for a detailed comparison of different object detection models). More precisely, the most recent YOLO versions YOLOv5 [JAS$^+$] and YOLOv8 [JCQ], are utilized.

Models based on the YOLOv5 architecture are composed of three main parts. The first one, often called "backbone", is a feature pyramid based on the CSP Darknet53 network [RF18]. The aggregated features of the backbone are fed into the "neck", which is a path aggregating network combining the features generated by the different stages of the backbone. Finally, the "head" performs the actual predictions: the bounding box prediction, class prediction, and confidence (objectness). YOLOv8 differs from YOLOv5 mostly by architectural changes (the "backbone" uses different convolutional layers, the "neck" uses different modules) and using other training losses [4].

[4] To the best of my knowledge, no papers describing the details of YOLOv5 or YOLOv8 have been published yet. To cite Glenn Jocher, the main contributor to both models: "YOLOv8 is not a published paper, but rather a series of improvements and extensions made by Ultralytics to the YOLOv5 architecture. Most of the changes made in YOLOv8 relate to model scaling and architecture tweaks, which can be found in the code and the documentation in the Ultralytics YOLOv8 repo [...]". Source: `https://github.com/ultralytics/ultralytics/issues/2572`, last accessed on 2023-09-03.

Both models, YOLOv5 and YOLOv8, are implemented and published for different network sizes, following the compound scaling method discussed in [TL19], where the network width, depth and resolution are scaled by fixed coefficients. This allows to adapt the network to the available data (e.g., fewer parameters would be better for smaller data sets), and on the size of the input images (e.g., more layers would be better for bigger input images). The smallest architecture is used for YOLOv5 (YOLOv5n), and the smallest two architectures for YOLOv8 (YOLOv8n and YOLOv8m).

The YOLOv5 achieves a mAP@50-95 of 34.3% on the validation split of images from the COCO data set [LMB+14] with a size of 640x640 pixels using the smallest network size (YOLOv5n); YOLOv8 achieves a mAP@50-95 of 37.3% on the same data and likewise the smallest network size (YOLOv8n). The comparably good performance by a short training time (2-3 days for training from scratch on the COCO data set – the data used within this thesis will be significantly smaller and pre-trained networks can be used), combined with the possibility to easily scale the network size, are the reasons why the YOLOv5 and YOLOv8 models are chosen as HTD models within this thesis.

2.4 Using Synthetic Handwritten Text

One approach for HTD based on synthetic data, combining base images with HWT, was already applied [JKSC20], where a method for segmenting handwritten and printed text at pixel level was evaluated. Documents from the PRImA layout analysis data set [ABPP09], and a collection of various paper forms are used as base images, and handwritten text from the IAM data set [MB02] is added to the documents. Basic augmentation techniques, like rotations or varying transparency, are applied on the HWT before merging with the base images. However, the HWT was randomly placed across the base images, which resulted in less realistic annotations compared to putting the text in content-free areas.

Fogel et al. [FAEC+20] evaluated their GAN-based image generation model by using the synthetic images for training a HTR model and successfully improved the performance on, among others, the IAM Handwriting Database [MB02] by augmenting the real training data with 100,000 generated images: the word error rate improved from 25.10% to 23.61%, the character error rate improved from 13.82% to 13.42%. Further, generating 100,000 images in the style and lexicon of the CVL Database [KFDS13], and training the HTR model on those synthetic and IAM images togehter, reduced the character error rate to 14.52%, compared to 15.62% when trained on the CVL data alone.

Notably, these error rates are still higher compared to the model presented in [CCP21], which was published in 2021 – the model used by Fogel et al., discussed in [BKL+19], was published in 2019. Further, while Fogel et al. use images on word-level, Coquenet et al. use entire line images. However, important is the insight that using synthetic data can improve HTR models, which is a major topic of this thesis.

CHAPTER 3

Methodology

As already outlined in Chapter 1, the focus of this thesis is on the generation of data sets using synthetic HWT. Existing and partially pre-trained models for HTD and HTR are trained on this data, and evaluated on real data. This chapter explains the methodology for generating the synthetic data sets.

Section 3.1 gives an overview for the actual usage of the HWT synthesis model from [DMP$^+$20], which is used for generating the HWT images needed for the synthetic data sets (see Chapter 2 for an explanation of its architecture). This is followed by a detailed description of the synthetic and evaluation data sets for HTD in Section 3.2. Section 3.3 gives an overview of the synthetic and evaluation data sets for HTR.

3.1 Handwritten Text Generation

As discussed in Chapter 2, two major approaches exist for synthetically generating handwritten text. The first one is predicting a sequence of coordinates in a two-dimensional space, which, after connecting them, can be interpreted as text. The second one generates images that already contain the text. For this work, the focus is laid on the second approach, especially on the state-of-the-art approach from [DMP$^+$20]. The main reason is its ability to mimic varying pen pressures, represented by varying color intensities of the generated text.

The GAN proposed by [DMP$^+$20] requires two inputs for generating images of handwritten text:

- The input text to show on the image: This can be any string containing characters in the character set the model was trained with. A pre-trained model with the IAM-dataset [MB02] as training data is used; hence the input string must follow the character set from the IAM database, which includes all alphanumerical characters from the English alphabet (a-z, A-Z, 0-9), the characters !"#&'()*+,-./:;?, and the space character.

- A style vector: The model from [DMP+20] uses a multivariate normal distributed latent space with 128 dimensions to represent the style of the handwritten text to mimic. Hence, a vector $\mathbf{v} \sim \mathcal{N}(\mathbf{0}_{128}, \mathbb{1}_{128})$ is needed as input as well.

Different text corpora are used as basis for generating the input strings:

LOTR *The Fellowship Of The Ring* is the first part of a fantasy novel by J. R. R. Tolkien, published in 1954 with over 180,000 words.

LOB The *Lancaster-Oslo/Bergen Corpus of British English* [SLG78], which is a collection of various British texts published in 1961. The corpus contains about 1 million words from 500 texts, each having about 2000 words, and was published in 1978. The IAM data set [MB02] is based on this text collection.

GUT This text corpus contains the books *Alice's Adventures in Wonderland* by Lewis Carroll, published in 1865, and *The Tragedie of Hamlet* by William Shakespeare, published in 1599. Both texts were taken from the *Project Gutenberg* text corpus from the Natural Language Toolkit [BKL09], a Python library for natural language processing.

Since the characters of the input string must be a subset of the character set the model was trained with, all invalid characters are removed from the text corpora. Further, all new line or tab characters are replaced by a space, and multiple occurrences of space characters are truncated to only one occurrence. Finally, random strings of varying lengths (uniformly distributed with data-set dependent boundaries, e.g. between 1 and 90 characters) are extracted from the processed text corpora such that no words are split up; those strings are the input strings for the generative model.

To mimic different styles of HWT, multiple style vectors are drawn from a random distribution, which is the multivariate standard normal distribution for most data sets. However, although the generative model was trained to have a normally distributed latent space, using vectors with extreme values still provides meaningful output, as already discussed in Chapter 2. Hence, style vectors following a uniform distribution over the interval $[-4, 4]$ are also used, as described in Section 3.3.

As shown in Figure 2.1, the model from [DMP+20] generates images that already mimic different HWT font styles. However, all images mimic roughly the same stroke width, and the model can only generate grayscale images. Hence, the stroke width and stroke color are additionally modified for some synthetic data sets (see Sections 3.2 and 3.3 for a detailed description of all synthetic data sets) to increase the diversity of styles: Image dilation is applied to increase the stroke width; The foreground of the generated images is computed and applied as fully transparent alpha channel on randomly colored images to mimic different colors. More details on both are available in Section 3.2.

3.2 Data Sets for Handwritten Text Detection

On a high level, existing images of magazines, newspapers, or scientific papers are extended with images of HWT to mimic annotated documents. Object detection models (YOLOv5 [JAS$^+$] and YOLOv8 [JCQ], see Section 2.3) are trained on this data to find the areas of handwritten text. The performance of this approach is evaluated by predicting areas with HWT on actual documents which contain such areas.

Section 3.2.1 outlines the approach for generating the synthetic data sets. Section 3.2.2 describes the synthetic data sets that are generated. Section 3.2.3 describes the actual data sets used for evaluations. The final evaluation results are described in Chapter 4.

3.2.1 Generating Synthetically Annotated Documents

Synthetic data sets for HTD are generated by adding synthetic images of HWT to "base images", which are scans of newspapers, magazines, or other documents. The HWT images are placed to mimic actual annotations, i.e., on areas where they do not overlap with any content (text, tables, images, separators, etc.) of the base images. A heuristic algorithm is applied for this purpose, which, on a high level, does following:

1. Image selection: A base image is randomly selected from the set of all possible base images.
2. Background area selection: A rectangular area within the base image, which only contains background, is randomly selected.
3. HWT paragraph creation: Images of HWT lines are vertically stacked to form a paragraph that fits into the selected background area. Modifications on the HWT line images are applied before stacking, especially regarding stroke color and stroke width, or image rotation.
4. HWT paragraph placement: The assembled HWT paragraph is placed on a random location within the selected background area. Some modifications on the paragraph image are applied before, such as randomly resizing or rotating the image.
5. Repeat steps 2. - 4. until a randomly chosen upper limit of paragraphs has been added or no suitable background area is found anymore[1].
6. Apply data-set dependent post-processing on the image: This is either a color-scale conversion (from RGB to gray-scale or binarized black-white images), or creation of cutouts of size 640x640 pixels to meet the requirements on the size of input images for YOLOv5, or both, or none of them.

[1] Each document has a randomly chosen number of 1 .. 12 paragraphs (those boundaries are empirically defined). The algorithm stops after 1000 iterations (also empirically defined), hence less paragraphs than this randomly chosen upper limit are possible as well.

7. Start again at step 1. until the desired number of synthetically annotated documents (data set dependent, most have multiple thousands of images) has been created.

The base images are taken from the PRImA-LAD data set [ABPP09], a layout analysis data set containing scans of over 400 pages from magazines or technical articles. Each scan is accompanied by a detailed description of the layout of the page (e.g. information about regions containing charts, images, noise, separators, text, or other content), allowing to distinguish foreground and background areas. 382 images with a width from 2080 – 4808 pixels, a height from 2858 – 3533 pixels, and a resolution of 300 DPI remain after sorting out the images with an incomplete layout description or containing areas with HWT. Those 382 scans are the base images for the synthetic HTD data sets; one such base image is possibly used multiple times to generate synthetic images with more than 382 items, but the synthetic annotations are added with a different random seed.

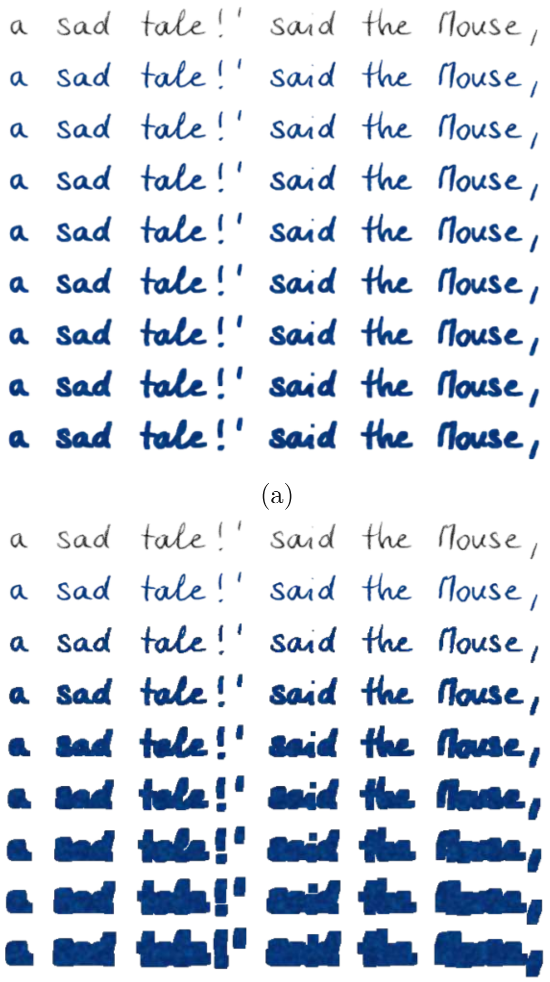
A synthetic paragraph consists of up to ten lines, the upper bound is randomly chosen and differs between paragraphs. All line images of one paragraph are based on the same style vector. From all such images, those that fit into the available background region are selected, taking the overall scaling factor of the entire paragraph into account. The lines are randomly rotated by ±1°[2] to mimic line orientations which are not exactly parallel. The background area is assumed to always have a “landscape” orientation when computing the maximum possible size of HWT lines, i.e. its width is greater or equal to its height. This ensures that vertically aligned rectangular areas are properly utilized.

Image dilation on upscaled HWT images is applied to mimic varying stroke widths to further augment the “originally” synthetic HWT images. The size of the dilation kernels is randomly chosen from 1x1, 2x2, 3x3, 5x5, 10x10, a kernel of size 1x1 means that no stroke width alteration was applied at all. All kernel sizes are applied with the same probability. If varying stroke widths are used for a data set, then all HWT line images within a paragraph are dilated with the same kernel size, but different paragraphs are possibly modified using different kernels. The effect of different image dilation kernel sizes, and the effect of upscaling the images before dilation, is visualized in Figure 3.1.

The HWT line images are artificially colored since the generative model for HWT images produces grayscale data only. This is done by extracting a foreground mask from the HWT images – i.e., a boolean mask indicating whether a pixel belongs to text – and applying this mask on another image, where foreground pixels are fully transparent. This other image is first randomly filled with selected colors from a certain color shade (i.e. various shades of red, blue, green, or dark gray). All HWT lines of one paragraph share the same color shade, but every paragraph on a document might have a different shade assigned. Further details about stroke color augmentation can be found in Appendix A.

The assembled paragraphs are rotated before they are pasted into the base imaged by $\alpha = (a + b)^\circ$, where a is randomly chosen from $0, 90, 180, 270$, and b is a random integer

[2] The value of ±1° is empirically defined to keep a balance between clearly visible line rotations and extreme line spacings for paragraphs with long lines.


a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,
a sad tale!' said the Mouse,

(a)

(b)

Figure 3.1: Effect of different dilation kernel sizes and image resizing on the simulated stroke width. Both sub-figures show the text “a sad tale!’ said the Mouse,” which is a random text passage from the *GUT* text corpus. The first and second lines are equal in both images. The first line shows an image generated by the model described in [DMP$^+$20], the second one is the same line but colored with shades of blue. All other lines show the second line but with an altered stroke width using a dilation kernel size of nxn, with n being 2, 4, 6, ..., 14. The lines in (a) have been resized by a factor of 4 (empirically defined) before applying the dilation operation; no resizing was done for the lines in (b). Notable is that resizing the images increases the granularity of the image dilation operation.

from the interval $[-3, 3]$. However, this random value is further bounded by the maximum number of degrees this paragraph can be rotated with to ensure that the paragraph does not overlap with any content. Details about the computation of this value is described in Appendix B. In case the selected background area has “portrait” orientation, i.e. it is higher than it is wide, the paragraph is rotated by 90° before further applying the rotation operation.

The paragraphs are randomly resized before placing it on the base image to mimic varying font sizes. The height of the individual image lines after resizing varies from 40 to 200 pixels. Assuming a document scanned with 300 DPI (which is the case for the base images from the PRImA-LAD data set), computer written fonts with a size between 12pt and 50pt would have a height of about 40 to 200 pixels. Notable is that the generated synthetic HWT images always have a height of 64 pixels, but the text does not always cover the entire height, hence the effective font size is possibly smaller than 64 pixels. However, this property is exploited when stacking the images vertically into a paragraph, as this causes a natural, style-dependent line spacing when vertically stacking the line images without additional padding.

To support training image sizes of 640x640 pixels (as, e.g., needed for object detection models YOLOv5), cutouts of that size with a stride of 160 pixels are created for some data sets (see Table 3.1). The images are downscaled by 50% before resizing to reduce the size of the data set.

The images of some data sets are converted from RGB to grayscale or binarized black-white images to control for the influence of different color scales on the prediction performance. Otsu’s adaptive thresholding method [Ots79] is utilized for latter.

The synthetic HTD data set labels are the bounding boxes of areas containing HWT for different granularities – entire paragraphs, lines, or single words. The bounding boxes for paragraphs are explicitly known. The position of a line within a paragraph is indirectly known; the absolute position can be derived from its relative position within a paragraph. However, the HWT synthesis model does not output any information about the location of words within a line, which must be computed additionally (see Appendix C). The absolute position of a word within the final image can be computed once the relative location of a word within a line is known.

3.2.2 Synthetic Data Sets for Handwritten Text Detection

Different data sets are generated to control for various aspects of their properties, especially:

- Color scales of the entire image data set
- HWT stroke width
- Label granularity

	Cutouts	Stroke Width Augmentation	Color Scale	Label Granularity
COLSCALES-RGB-CUT-PAR	✓	✗	RGB	P
COLSCALES-GRAY-CUT-PAR	✓	✗	GRAY	P
COLSCALES-BW-CUT-PAR	✓	✗	BW	P
COLSCALES-STROKE-BW-CUT-PAR	✓	✓	BW	P
FULLSIZE-STROKE-BW-PAR	✗	✓	BW	P
GRANULARITY-STROKE-BW-PAR	✗	✓	BW	P
GRANULARITY-STROKE-BW-LINE	✗	✓	BW	L
GRANULARITY-STROKE-BW-WORD	✗	✓	BW	W
CWT-STROKE-BW-PAR	✗	✓	BW	P
CWT-STROKE-BW-LINE	✗	✓	BW	L
CWT-STROKE-BW-WORD	✗	✓	BW	W

Table 3.1: A summary of different properties of the synthetic data sets for HTD. Synthetic paragraphs contain not more than ten lines with a maximum font size of 200 pixels. The following abbreviations are used: gray scale (GRAY), black-white (BW), paragraph (P), line (L), word (W).

All data sets are split up into a training, validation, and test set. The HWT detection models are trained on the images from the training set only. The validation set is used for early stopping. All results presented in Chapter 4 are based on the test set. The data is split on base image level, i.e., all synthetically annotated documents in the training, validation and test sets are based on the same base images for all data sets. The synthetic data sets and their configurations are listed in Table 3.1. The number of images and total number of objects for each data set are listed in Table 3.2 and Table 3.3, respectively.

The *COLSCALES-** data sets are all based on 2,000 synthetically annotated documents, hence the base images from the PRImA-LAD data set occur on average more than 5 times. The actual training data is based on cutouts of the images with size 640x640 pixels and a stride of 160 pixels. The images are downscaled by 50% before creating the cutouts to reduce the total number of images. The training, validation and test sets contain 32,814, 7,380 and 6,335 images, respectively. Stroke color augmentation is applied to the HWT before adding it to the base images. The font size is between 40 and 200 pixels, with up to ten lines per paragraph. *COLSCALES-RGB-CUT-PAR*

	# Training Images	# Validation Images	# Test Images
COLSCALES-RGB-CUT-PAR	32,814	7,380	6,335
COLSCALES-GRAY-CUT-PAR	32,814	7,380	6,335
COLSCALES-BW-CUT-PAR	32,814	7,380	6,335
COLSCALES-STROKE-BW-CUT-PAR	32,946	7,353	6,398
FULLSIZE-STROKE-BW-PAR	1,412	312	276
GRANULARITY-STROKE-BW-PAR	1,401	309	275
GRANULARITY-STROKE-BW-LINE	1,401	309	275
GRANULARITY-STROKE-BW-WORD	1,401	309	275
CWT-STROKE-BW-PAR	2,401	309	275
CWT-STROKE-BW-LINE	2,401	309	275
CWT-STROKE-BW-WORD	2,401	309	275

Table 3.2: A summary about the number of images for the synthetic data sets for HTD.

contains RGB images, *COLSCALES-GRAY-CUT-PAR* contains only grayscale images, and *COLSCALES-BW-CUT-PAR* only binarized data. All labels are on paragraph level (or cutouts thereof).

Further analyzing the synthetic images reveals that the stroke width of HWT text was generally relatively thin compared to real-world images, e.g., from the CVL data set [KFDS13]. Hence, stroke width augmentation is introduced for the data set *COLSCALES-STROKE-BW-CUT-PAR*, which is equal to *COLSCALES-RGB-CUT-PAR* with all other parameters.

Switching to a different object detection model (YOLOv8 vs. YOLOv5, see Chapter 2 for further information) allows using bigger input images. Hence, creating cutouts from the synthetic images is not needed anymore; the images can be used as-is, as done for the data set *FULLSIZE-STROKE-BW-PAR*, which is equal to *COLSCALES-STROKE-BW-CUT-PAR* except no cutouts are generated.

Generating synthetically annotated documents allows full control over the granularity of the ground truth, i.e., about the position of paragraphs, lines, or single words. The data sets *GRANULARITY-STROKE-BW-PAR*, *GRANULARITY-STROKE-BW-LINE*, and *GRANULARITY-STROKE-BW-WORD* all have equal configuration, but labels on paragraph, line, and word level, respectively. Further, those data sets additionally have an equal configuration to *FULLSIZE-STROKE-BW-PAR*, but the final images appear differently due to changes in the internal random states since they are generated after

	# Objects Training Set	# Objects Validation Set	# Objects Test Set
COLSCALES-RGB-CUT-PAR	90,659	19,506	17,940
COLSCALES-GRAY-CUT-PAR	90,659	19,506	17,940
COLSCALES-BW-CUT-PAR	90,659	19,506	17,940
COLSCALES-STROKE-BW-CUT-PAR	91,102	19,336	17,947
FULLSIZE-STROKE-BW-PAR	17,879	3,930	3,484
GRANULARITY-STROKE-BW-PAR	17,544	3,890	3,436
GRANULARITY-STROKE-BW-LINE	22,779	4,992	4,566
GRANULARITY-STROKE-BW-WORD	91,032	20,341	18,281
CWT-STROKE-BW-PAR	20,515	3,890	3,436
CWT-STROKE-BW-LINE	30,550	4,992	4,566
CWT-STROKE-BW-WORD	139,605	20,341	18,281

Table 3.3: A summary of the number of objects (i.e. the number of synthetic HWT paragraphs, lines or words) for the synthetic data sets for HTD.

modifications of the underlying algorithm introducing the computation of bounding boxes for lines and words.

The base images from the PRImA-LAD collection usually have the content centered within the image - often using a two-column layout. Hence, the generated images have most of the HWT content towards the border of the documents. To account for that, i.e., to break up that structure by generating additional images with a more diverse distribution of HWT content, different kinds of synthetic documents are added to the data set: Paragraphs based on the *LOTR* corpus using computer fonts, and synthetic HWT text paragraphs, are randomly placed on empty images, which have a width and height randomly chosen and limited by the minimum and maximum width and height of the images from the PRImA-LAD data set. The data sets *CWT-STROKE-BW-PAR*, *CWT-STROKE-BW-LINE*, and *CWT-STROKE-BW-WORD* are equal to *GRANULARITY-STROKE-BW-**, except their training data is extended with 1000 of such synthetic images.

Figure 3.2 shows a synthetically annotated document in RGB color scale, similar to the images of the *COLSCALES-RGB-CUT-PAR* data set before creating cutouts (however, stroke width augmentation is applied for the image in Figure 3.2). A sample of a binarized image, similar to those from the *FULLSIZE-STROKE-BW-PAR* data set, is shown in

Figure 3.3. Figure 3.4 contains an example for the fully synthetic image with random CWT and HWT paragraphs.

3.2.3 Evaluation Data Sets for Handwritten Text Detection

Object detection models are trained on the data sets described in Section 3.2.2 to distinguish areas of HWT from everything else in the documents (CWT, images, and more). The performance of those models is evaluated using two data sets.

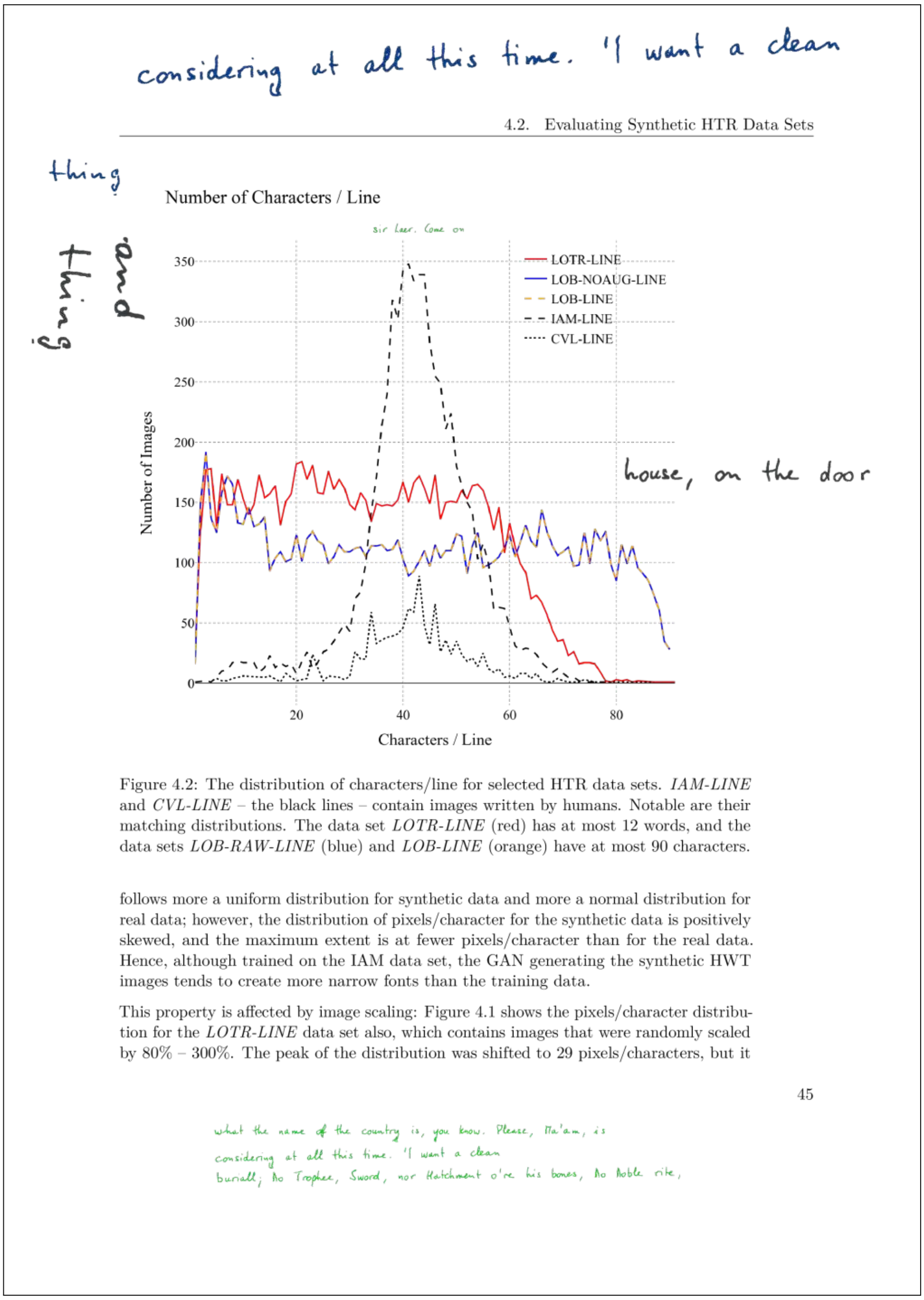
The first one is the CVL-Database [KFDS13], which contains in total 1604 scanned pages with CWT and HWT. The images contain an area of CWT on the top, which are transcribed by, in total, 310 participants beneath this area. Seven different texts were to transcribe, of which one was in German, the others in English. The German text and one of the English texts are not used within this work, as they contain German umlauts which the HWT synthesis model was not trained to generate. The data set is split into 189 training images and 1415 test images. All images are accompanied by a ground truth containing the bounding boxes and content of the entire HWT area, single lines, and individual words. The scans are available in an RGB color scale, but most HWT was written with a blue pen. An example image of the CVL data set is shown in Figure 3.5. Further summary of the properties of the CVL data set are shown in Table 3.4. Not all data sets based on the CVL database are explicitly listed, but the following rules apply: the name of all data sets based on the CVL database start with *CVL-*; those ending with *-PAR*, *-LINE* or *-WORD* contain labels on the paragraph-, line- or word-level, respectively; those that contain *-BW-* or *-GRAY-* contain binarized or grayscale images, respectively; the images of those data sets with a *-CUT-* in the name were downscaled by 50% and contain cutouts of size 640x640 pixels with a stride of 160 pixels.

The second data set was created specifically for this thesis. It contains a scan of the paper "ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation" [FAEC+20], which was manually annotated by two persons with random text lines from [Tol20]. Ten images are available in total. The annotations were made with different pens and pen colors. Each image contains various annotations, most of which contain multiple lines. Labels were created manually for paragraph-, line- and word-level by one person using the web-based image annotation software DataTorch[3]. Figure 3.6 shows an example image of this data set. Further details can be found in Table 3.4. The same naming rules which apply to the *CVL-** data sets also apply to the *SCAN-** data sets.

3.3 Data Sets for Handwritten Text Recognition

Models for handwritten text recognition take images of HWT as input and predict a sequence of characters shown in the image at hand. The focus of this thesis is the use of synthetic training images. Thus, Section 3.3.1 outlines the approach for generating such

[3] https://datatorch.io, last accessed on 2023-07-16



considering at all this time. 'I want a clean

thing

thing

and

4.2. Evaluating Synthetic HTR Data Sets

house, on the door

Figure 4.2: The distribution of characters/line for selected HTR data sets. *IAM-LINE* and *CVL-LINE* – the black lines – contain images written by humans. Notable are their matching distributions. The data set *LOTR-LINE* (red) has at most 12 words, and the data sets *LOB-RAW-LINE* (blue) and *LOB-LINE* (orange) have at most 90 characters.

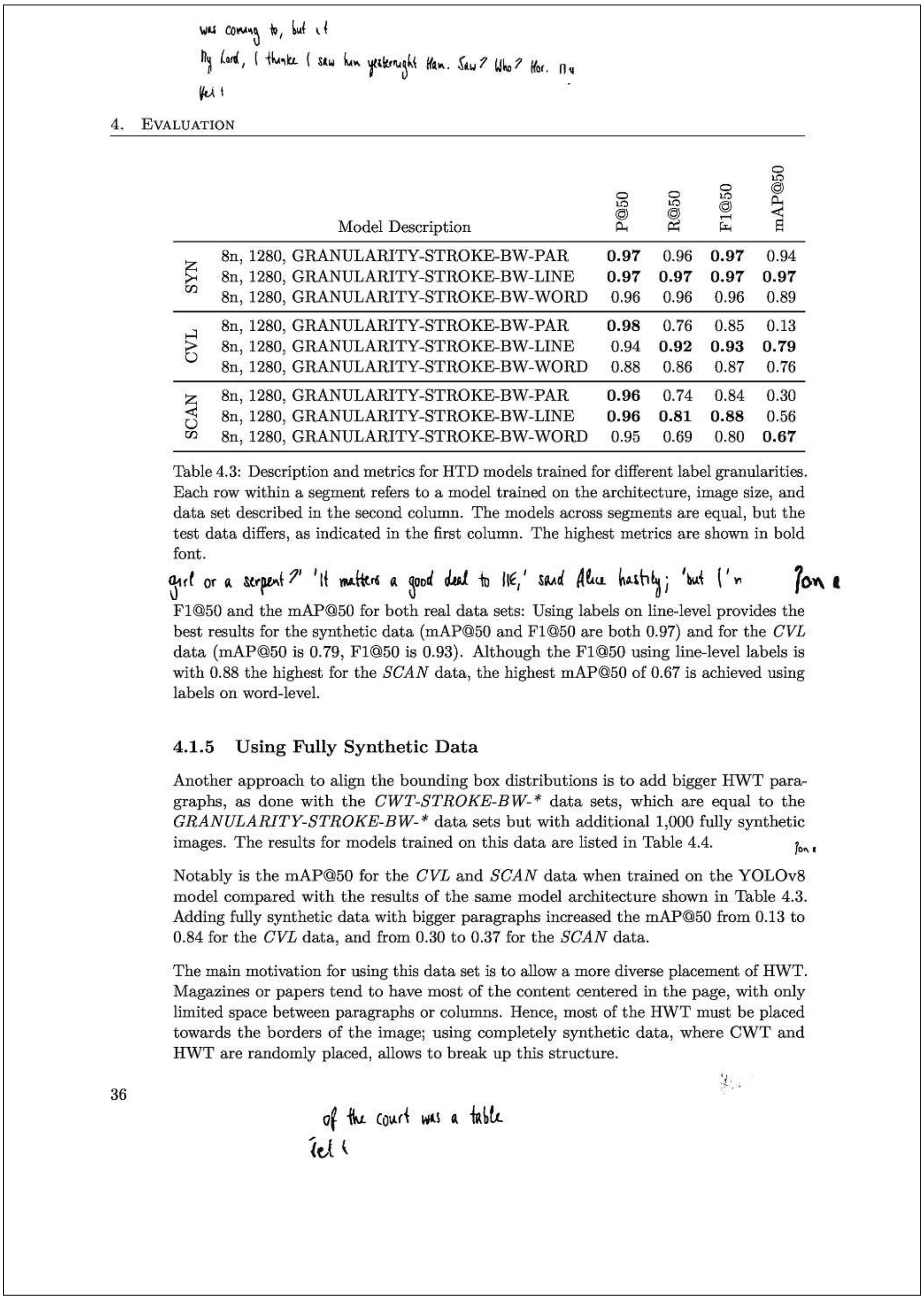
follows more a uniform distribution for synthetic data and more a normal distribution for real data; however, the distribution of pixels/character for the synthetic data is positively skewed, and the maximum extent is at fewer pixels/character than for the real data. Hence, although trained on the IAM data set, the GAN generating the synthetic HWT images tends to create more narrow fonts than the training data.

This property is affected by image scaling: Figure 4.1 shows the pixels/character distribution for the *LOTR-LINE* data set also, which contains images that were randomly scaled by 80% – 300%. The peak of the distribution was shifted to 29 pixels/characters, but it

45

what the name of the country is, you know. Please, Ma'am, is
considering at all this time. 'I want a clean
buriall; No Trophee, Sword, nor Hatchment o're his bones, No Noble rite,

Figure 3.2: An example of a synthetically annotated document, similar to the images from the *COLSCALES-RGB-CUT-PAR* data set before creating cutouts (but with stroke width augmentation). A page from this thesis is taken as base image, the synthetic HWT is generated with the model proposed by [DMP^{+}20] using input strings from the *GUT* text corpus.



was coming to, but it

My Lord, I thinke I saw him yesternight Han. Saw? Who? Hor. My

Yei t

4. Evaluation

	Model Description	P@50	R@50	F1@50	mAP@50
SYN	8n, 1280, GRANULARITY-STROKE-BW-PAR	**0.97**	0.96	**0.97**	0.94
	8n, 1280, GRANULARITY-STROKE-BW-LINE	**0.97**	**0.97**	**0.97**	**0.97**
	8n, 1280, GRANULARITY-STROKE-BW-WORD	0.96	0.96	0.96	0.89
CVL	8n, 1280, GRANULARITY-STROKE-BW-PAR	**0.98**	0.76	0.85	0.13
	8n, 1280, GRANULARITY-STROKE-BW-LINE	0.94	**0.92**	**0.93**	**0.79**
	8n, 1280, GRANULARITY-STROKE-BW-WORD	0.88	0.86	0.87	0.76
SCAN	8n, 1280, GRANULARITY-STROKE-BW-PAR	**0.96**	0.74	0.84	0.30
	8n, 1280, GRANULARITY-STROKE-BW-LINE	**0.96**	**0.81**	**0.88**	0.56
	8n, 1280, GRANULARITY-STROKE-BW-WORD	0.95	0.69	0.80	**0.67**

Table 4.3: Description and metrics for HTD models trained for different label granularities. Each row within a segment refers to a model trained on the architecture, image size, and data set described in the second column. The models across segments are equal, but the test data differs, as indicated in the first column. The highest metrics are shown in bold font.

girl or a serpent?' 'It matters a good deal to ME,' said Alice hastily; 'but I'm Pon a

F1@50 and the mAP@50 for both real data sets: Using labels on line-level provides the best results for the synthetic data (mAP@50 and F1@50 are both 0.97) and for the *CVL* data (mAP@50 is 0.79, F1@50 is 0.93). Although the F1@50 using line-level labels is with 0.88 the highest for the *SCAN* data, the highest mAP@50 of 0.67 is achieved using labels on word-level.

4.1.5 Using Fully Synthetic Data

Another approach to align the bounding box distributions is to add bigger HWT paragraphs, as done with the *CWT-STROKE-BW-** data sets, which are equal to the *GRANULARITY-STROKE-BW-** data sets but with additional 1,000 fully synthetic images. The results for models trained on this data are listed in Table 4.4. Pon

Notably is the mAP@50 for the *CVL* and *SCAN* data when trained on the YOLOv8 model compared with the results of the same model architecture shown in Table 4.3. Adding fully synthetic data with bigger paragraphs increased the mAP@50 from 0.13 to 0.84 for the *CVL* data, and from 0.30 to 0.37 for the *SCAN* data.

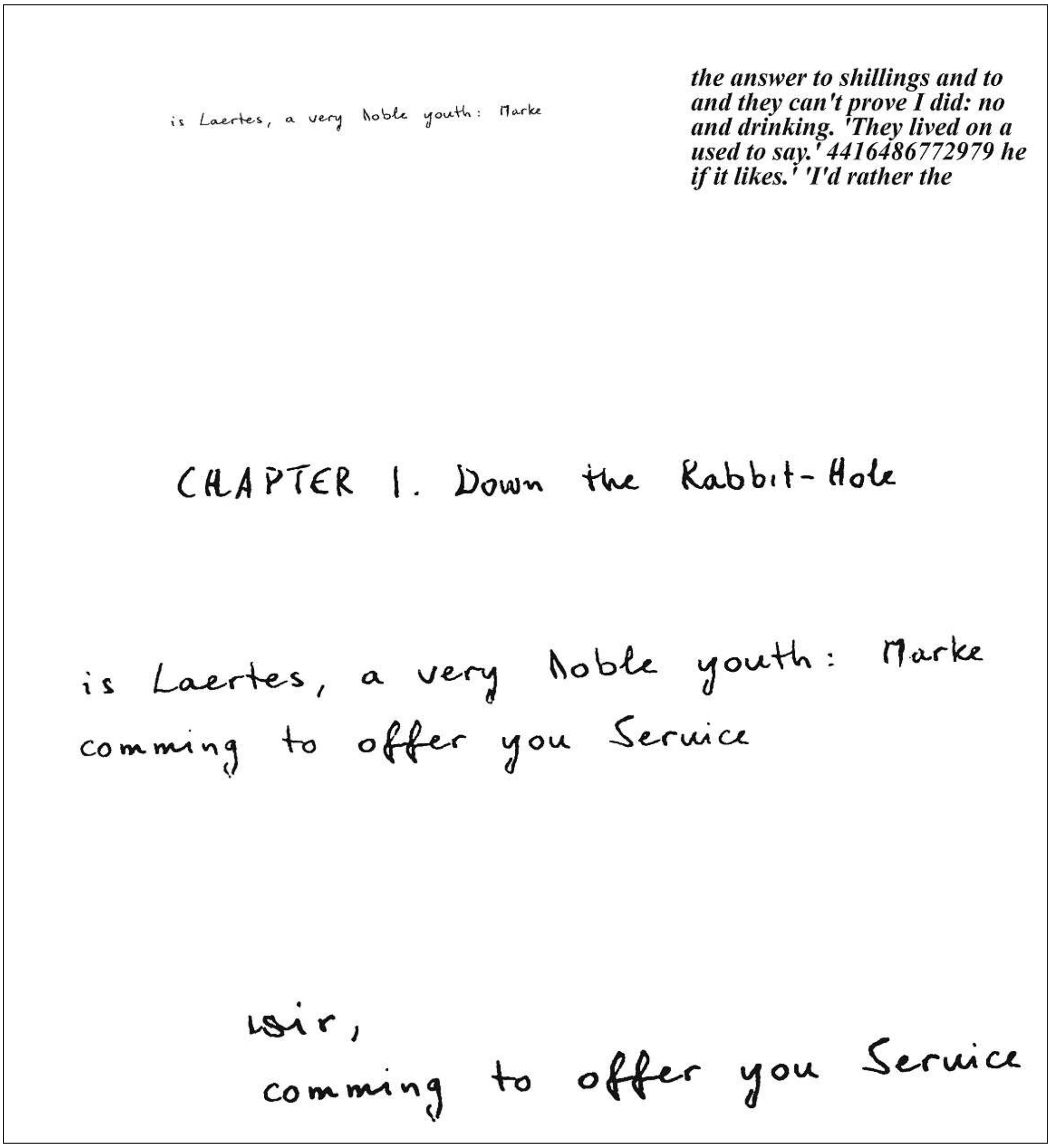
The main motivation for using this data set is to allow a more diverse placement of HWT. Magazines or papers tend to have most of the content centered in the page, with only limited space between paragraphs or columns. Hence, most of the HWT must be placed towards the borders of the image; using completely synthetic data, where CWT and HWT are randomly placed, allows to break up this structure.

36

of the court was a table

Tel (

Figure 3.3: An example of a synthetically annotated and binarized document, similar to the images from the *FULLSIZE-STROKE-BW-PAR* data set. A page from this thesis is taken as base image, the synthetic HWT is generated with the model proposed by [DMP$^+$20] using input strings from the *GUT* text corpus.



is Laertes, a very Noble youth: Marke

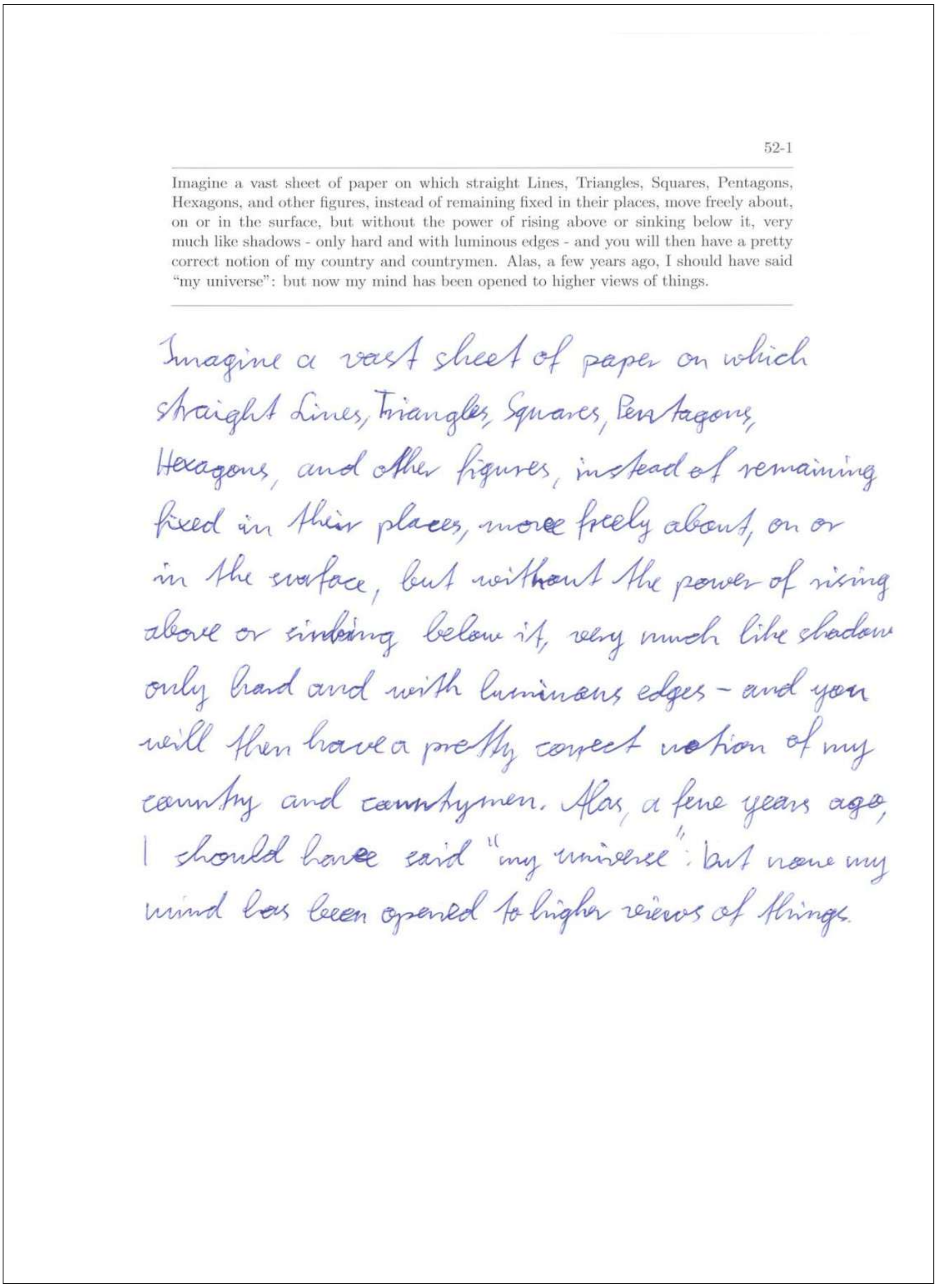
the answer to shillings and to
and they can't prove I did: no
and drinking. 'They lived on a
used to say.' 4416486772979 he
if it likes.' 'I'd rather the

CHAPTER I. Down the Rabbit-Hole

is Laertes, a very Noble youth: Marke
comming to offer you Seruice

wir,
comming to offer you Seruice

Figure 3.4: An example of a fully synthetic document, similar to the images of the *CWT-STROKE-BW-PAR* data set. Artificially assembled paragraphs using CWT and synthetic HWT (generated using the model proposed by [DMP+20]) are randomly added to an empty image. The text samples are taken from the *GUT* text corpus, but extended with random alpha-numeric strings.

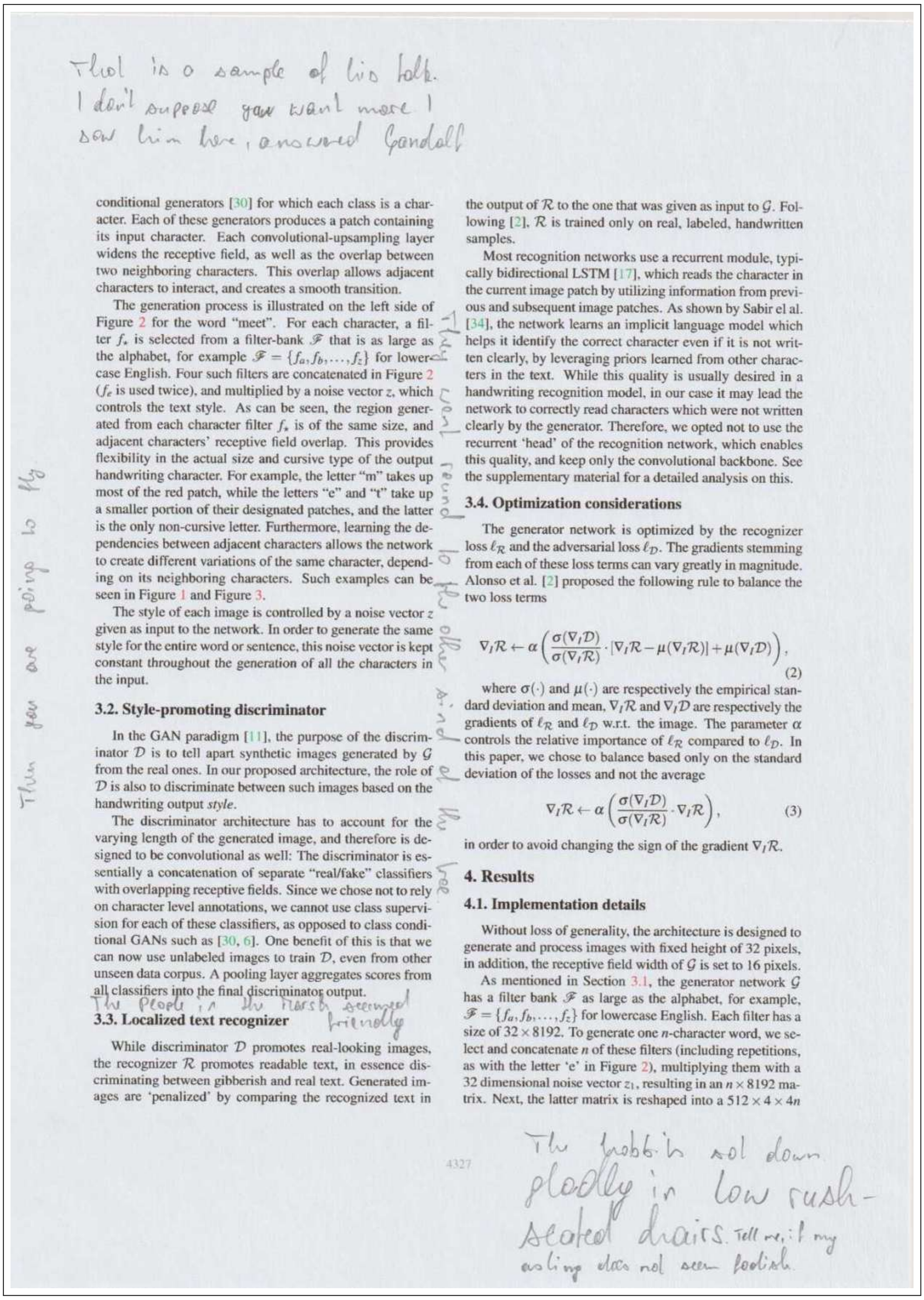


52-1

Imagine a vast sheet of paper on which straight Lines, Triangles, Squares, Pentagons, Hexagons, and other figures, instead of remaining fixed in their places, move freely about, on or in the surface, but without the power of rising above or sinking below it, very much like shadows - only hard and with luminous edges - and you will then have a pretty correct notion of my country and countrymen. Alas, a few years ago, I should have said "my universe": but now my mind has been opened to higher views of things.

Imagine a vast sheet of paper on which
straight Lines, Triangles, Squares, Pentagons,
Hexagons, and other figures, instead of remaining
fixed in their places, move freely about, on or
in the surface, but without the power of rising
above or sinking below it, very much like shadow
only hard and with luminous edges - and you
will then have a pretty correct notion of my
country and countrymen. Alas, a few years ago,
I should have said "my universe": but now my
mind has been opened to higher views of things.

Figure 3.5: A sample image from the CVL data base (`0052-1.tif` from the `testset`).



That is a sample of his talk. I don't suppose you want more I saw him here, answered Gandalf

Then you are going to fly

conditional generators [30] for which each class is a character. Each of these generators produces a patch containing its input character. Each convolutional-upsampling layer widens the receptive field, as well as the overlap between two neighboring characters. This overlap allows adjacent characters to interact, and creates a smooth transition.

The generation process is illustrated on the left side of Figure 2 for the word "meet". For each character, a filter f_* is selected from a filter-bank $\mathscr{F}$ that is as large as the alphabet, for example $\mathscr{F} = \{f_a, f_b, \dots, f_z\}$ for lowercase English. Four such filters are concatenated in Figure 2 (f_e is used twice), and multiplied by a noise vector z, which controls the text style. As can be seen, the region generated from each character filter f_* is of the same size, and adjacent characters' receptive field overlap. This provides flexibility in the actual size and cursive type of the output handwriting character. For example, the letter "m" takes up most of the red patch, while the letters "e" and "t" take up a smaller portion of their designated patches, and the latter is the only non-cursive letter. Furthermore, learning the dependencies between adjacent characters allows the network to create different variations of the same character, depending on its neighboring characters. Such examples can be seen in Figure 1 and Figure 3.

The style of each image is controlled by a noise vector z given as input to the network. In order to generate the same style for the entire word or sentence, this noise vector is kept constant throughout the generation of all the characters in the input.

3.2. Style-promoting discriminator

In the GAN paradigm [11], the purpose of the discriminator $\mathcal{D}$ is to tell apart synthetic images generated by $\mathcal{G}$ from the real ones. In our proposed architecture, the role of $\mathcal{D}$ is also to discriminate between such images based on the handwriting output *style*.

The discriminator architecture has to account for the varying length of the generated image, and therefore is designed to be convolutional as well: The discriminator is essentially a concatenation of separate "real/fake" classifiers with overlapping receptive fields. Since we chose not to rely on character level annotations, we cannot use class supervision for each of these classifiers, as opposed to class conditional GANs such as [30, 6]. One benefit of this is that we can now use unlabeled images to train $\mathcal{D}$, even from other unseen data corpus. A pooling layer aggregates scores from all classifiers into the final discriminator output.

The people in the marsh seemed friendly

3.3. Localized text recognizer

While discriminator $\mathcal{D}$ promotes real-looking images, the recognizer $\mathcal{R}$ promotes readable text, in essence discriminating between gibberish and real text. Generated images are 'penalized' by comparing the recognized text in the output of $\mathcal{R}$ to the one that was given as input to $\mathcal{G}$. Following [2], $\mathcal{R}$ is trained only on real, labeled, handwritten samples.

They went round to the other side of the tree

Most recognition networks use a recurrent module, typically bidirectional LSTM [17], which reads the character in the current image patch by utilizing information from previous and subsequent image patches. As shown by Sabir el al. [34], the network learns an implicit language model which helps it identify the correct character even if it is not written clearly, by leveraging priors learned from other characters in the text. While this quality is usually desired in a handwriting recognition model, in our case it may lead the network to correctly read characters which were not written clearly by the generator. Therefore, we opted not to use the recurrent 'head' of the recognition network, which enables this quality, and keep only the convolutional backbone. See the supplementary material for a detailed analysis on this.

3.4. Optimization considerations

The generator network is optimized by the recognizer loss $\ell_{\mathcal{R}}$ and the adversarial loss $\ell_{\mathcal{D}}$. The gradients stemming from each of these loss terms can vary greatly in magnitude. Alonso et al. [2] proposed the following rule to balance the two loss terms

$$\nabla_I \mathcal{R} \leftarrow \alpha \left(\frac{\sigma(\nabla_I \mathcal{D})}{\sigma(\nabla_I \mathcal{R})} \cdot [\nabla_I \mathcal{R} - \mu(\nabla_I \mathcal{R})] + \mu(\nabla_I \mathcal{D}) \right), \tag{2}$$

where $\sigma(\cdot)$ and $\mu(\cdot)$ are respectively the empirical standard deviation and mean, $\nabla_I \mathcal{R}$ and $\nabla_I \mathcal{D}$ are respectively the gradients of $\ell_{\mathcal{R}}$ and $\ell_{\mathcal{D}}$ w.r.t. the image. The parameter α controls the relative importance of $\ell_{\mathcal{R}}$ compared to $\ell_{\mathcal{D}}$. In this paper, we chose to balance based only on the standard deviation of the losses and not the average

$$\nabla_I \mathcal{R} \leftarrow \alpha \left(\frac{\sigma(\nabla_I \mathcal{D})}{\sigma(\nabla_I \mathcal{R})} \cdot \nabla_I \mathcal{R} \right), \tag{3}$$

in order to avoid changing the sign of the gradient $\nabla_I \mathcal{R}$.

4. Results

4.1. Implementation details

Without loss of generality, the architecture is designed to generate and process images with fixed height of 32 pixels, in addition, the receptive field width of $\mathcal{G}$ is set to 16 pixels.

As mentioned in Section 3.1, the generator network $\mathcal{G}$ has a filter bank $\mathscr{F}$ as large as the alphabet, for example, $\mathscr{F} = \{f_a, f_b, \dots, f_z\}$ for lowercase English. Each filter has a size of 32×8192. To generate one n-character word, we select and concatenate n of these filters (including repetitions, as with the letter 'e' in Figure 2), multiplying them with a 32 dimensional noise vector z_1, resulting in an $n \times 8192$ matrix. Next, the latter matrix is reshaped into a $512 \times 4 \times 4n$

4327

The hobbits sat down gladly in low rush-seated chairs. Tell me, if my asking does not seem foolish.

Figure 3.6: A sample image from the *SCAN-** data sets (page 4 of [FAEC$^+$20]) with manual annotations from [Tol20].

Property	CVL	SCAN
Image width	2,480 – 2,663 pixels	2,473 pixels
Image height	3,507 – 3,634 pixels	3,495 pixels
Number of images	1,411	10
Number of objects, total	1,409 / 11,849 / 88,825	65 / 228 / 877
Number of objects / image, average	1 / 8 / 63	6 / 23 / 88

Table 3.4: Summary of the evaluation data sets for HTD. If multiple values are listed in a table cell, then they refer to the paragraph-, line- and word-level data sets, respectively.

data sets, Section 3.3.2 describes the ones that have been created. Section 3.3.3 describes the real data sets used for evaluations. The final evaluation results are described in Chapter 4.

3.3.1 Generating Synthetic Data Sets for Handwritten Text Recognition

Two different kinds of data sets are generated, containing either paragraphs or lines only. The approach for creating synthetic paragraphs is similar to the one used for synthetic HTD data sets: synthetic images of HWT lines are vertically stacked after they have been modified (stroke width, stroke color, font size, indentation, rotation, and background removal). The generation of synthetic line data sets is a bit simpler, as only the line images themselves have to be modified (stroke width, stroke color, and font size).

3.3.2 Synthetic Data Sets for Handwritten Text Recognition

As for HTD, different data sets are generated to control for the influence of various aspects of their properties, especially:

- The text corpus
- Image scaling
- Data set size
- Style vectors for the HWT synthesis model

All data sets are split up into a training and test set, which have the same usage as for the HTD tasks. A validation set is not used, since no early stopping is applied while training the HTR models. A summary about the most important properties of the synthetic HTR data sets follows below and in Table 3.5.

The input strings for the first data set, *LOTR-LINE*, are based on the *LOTR* text corpus and were constrained to contain at most 12 words (basically a randomly chosen value as a basis for further analysis). Stroke width and stroke color augmentation are applied.

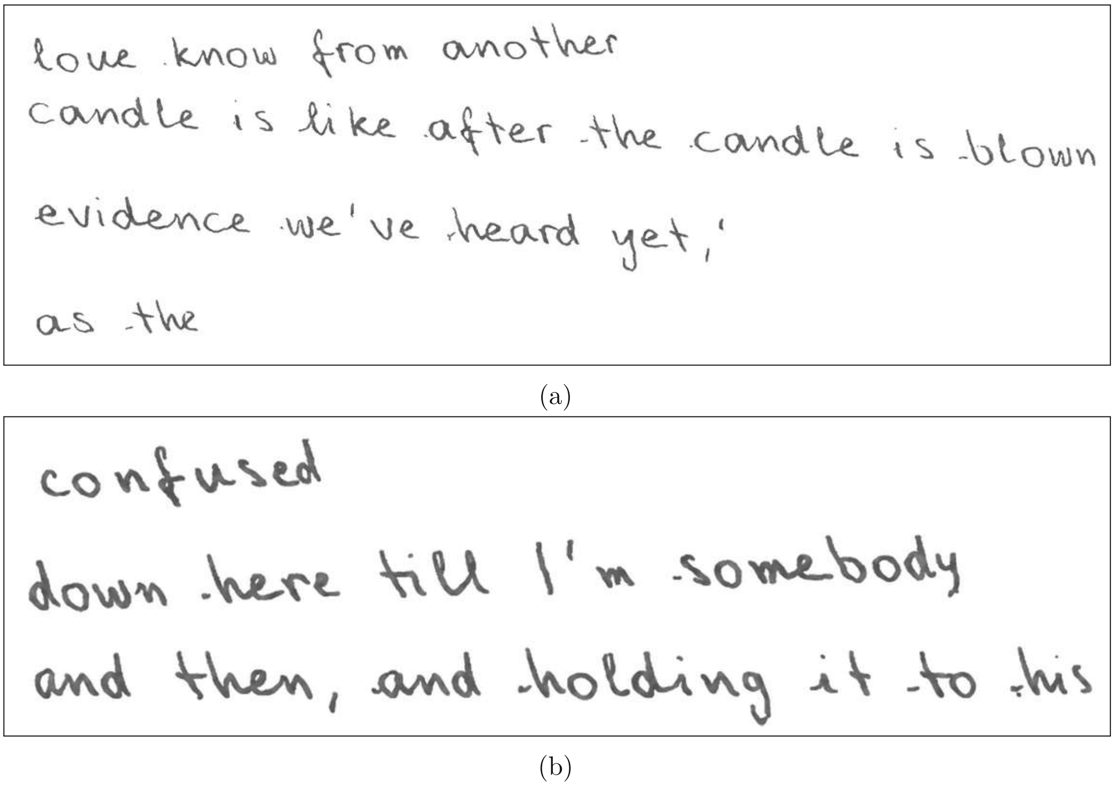
	Text Corpus	Content	Min. Font Size	Max. Font Size	Stroke Augmentations	# Training Images	# Test Images
LOTR-LINE	LOTR	Line	51	192	✓	10,000	2,500
GUT-PAR	GUT	Paragraph	51	70	✓	6,000	1,500
GUT-LINE	GUT	Line	51	70	✓	10,000	2,500
LOB-LINE	LOB	Line	39	300	✓	10,000	2,500
LOB-50K-LINE	LOB	Line	39	300	✓	50,000	5,000
LOB-UNIFORM-LINE	LOB	Line	39	300	✓	10,000	2,500
LOB-SEED-LINE	LOB	Line	39	300	✓	10,000	2,500
LOB-RAW-LINE	LOB	Line	64	64	✗	10,000	2,500
LOB-NOAUG-LINE	LOB	Line	64	64	✗	10,000	2,500

Table 3.5: A summary about different properties of the synthetic data sets for HTR.

To have a variation in word length and out-of-vocabulary words, the remaining data sets are based on different text corpora. One of them is the *GUT* text corpus, which is the basis for the *GUT-LINE* and *GUT-PAR* data sets. The former is a line data, the latter is a paragraph data set where between two and ten randomly chosen lines from the same style are stacked after some pre-processing was applied (stroke width and stroke color modification, line image rotation and indentation). Since training using a synthetic paragraph data set with up to 12 words per line and up to ten lines per paragraph exceeds the hardware resources (esp. available GPU memory on a NVIDIA A40), the lines for both data sets have at most 45 characters (which is about half of the maximum characters per line for the IAM data set [MB02]), ensuring that no words are split up, and the scaling factor for the lines in the data set is changed from 80% – 300% to 80% – 110% (i.e., maximum font size is limited to 70 pixels).

The input lines for the remaining six synthetic data sets are taken from the *LOB* text collection, which is the same corpus the IAM data set is based on, which is the data set the HWT generation model used within this thesis is trained on. A different image scaling strategy is applied to generate a more diverse distribution of pixels / character (see Section 4.2.3 for further details): the aspect ratio is distorted intentionally, and the images are scaled so that the widths after scaling more closely match the image width distribution of the IAM data set.

The following five data sets are based on this scaling strategy: *LOB-LINE*, *LOB-LINE-50K* containing 50k training images (vs. 10k as for the reminaing line data sets), *LOB-LINE-SEED* generated with a different initial random seed compared to *LOB-*


love know from another
candle is like after the candle is blown
evidence we've heard yet,'
as the
confused
down here till I'm somebody
and then, and holding it to his

(a)

(b)

Figure 3.7: Two sample images from the *GUT-PAR* data set. Synthetic HWT lines (generated using the model proposed by [DMP$^+$20]) are rotated, and vertically stacked. The text samples are taken from the *GUT* text corpus. No stroke width augmentation is applied for (a), opposed to (b).

LINE, *LOB-LINE-NOAUG* without any stroke augmentation (only image scaling), and *LOB-LINE-UNIFORM* where style vectors for the synthetic images are drawn from a multivariate uniform distribution over the interval $[-4, 4]$ instead of a multivariate standard normal distribution. The last data set, *LOB-RAW-LINE*, is likewise based on the *LOB* text corpus, but uses the raw data as-is without any modifications (no stroke augmentation, no image scaling).

Summarized, following synthetic HTR data sets are generated:

LOTR-LINE Line data set, based on the *LOTR* text corpus, up to 12 words per line. Contains stroke width and stroke color augmentation. Images are resized to a font size of 51 - 192 pixels.

GUT-PAR Paragraph data set, based on the *GUT* text corpus, up to 45 characters per line. Contains stroke width and stroke color augmentation. Images are resized to a font size of 51 - 70 pixels.

Property	IAM	CVL
Number of samples, train split	747 / 6,482	135 / 1,130
Number of samples, test split	336 / 2,915	846 / 6,836
Average number of characters / sample, train split	378 / 43	359 / 42
Average number of characters / sample, test split	374 / 42	350 / 42

Table 3.6: Summary of the evaluation data sets for HTR. Multiple values in a table cell describe the paragraph- and line-level data sets, respectively.

GUT-LINE Paragraph data set, based on the *GUT* text corpus, up to 45 characters per line. Contains stroke width and stroke color augmentation. Images are resized to a font size of 51 - 70 pixels.

LOB-LINE Line data set, based on the *LOB* text corpus, up to 90 characters per line. Contains stroke width and stroke color augmentation. Images are resized to a font size of 39 - 300 pixels, intentionally discarding the aspect ratio.

LOB-50K-LINE As *LOB-LINE*, but more images. The number of different styles for the generated synthetic HWT lines is increased as well, from 400 to 2,000 (with 25 images per style).

LOB-UNIFORM-LINE As *LOB-LINE*, but the style vectors for the synthetic HWT lines are drawn from a uniform distribution over the interval $[-4, 4]$ instead of a multivarate standard normal distribution.

LOB-SEED-LINE As *LOB-LINE*, but the random number generator for the style vectors for the synthetic HWT lines was initialized with a different seed.

LOB-RAW-LINE As *LOB-LINE*, but neither stroke width and stroke color augmentation is done, nor are the images resized.

LOB-NOAUG-LINE As *LOB-LINE*, but neither stroke width and stroke color augmentation is done, only image scaling is applied.

3.3.3 Evaluation Data Sets for Handwritten Text Recognition

Two data sets are used for evaluating the HTR model performance. The first one is the IAM Handwriting Database [MB02], which consists of 1,593 handwritten paragraphs (13,353 lines, 115,320 words) from 657 different persons. The text samples are based on the *LOB* text corpus. The second evaluation data set is the CVL database [KFDS13], already introduced in Section 3.2.3 - as for the HTD evaluation, only the texts without German umlauts are used, i.e. two texts are ignored. Images containing invalid HWT data (e.g. drawings) or labels are removed as well. Finally, all images are converted to grayscale. Table 3.6 summarizes the properties of both data sets.

The HTR model is trained with a specific character set, i.e., all characters within this set can be classified. The HTR model proposed in [CCP21], which is used for this thesis, was trained on the character set from the IAM database. The evaluation data based on the CVL data set does not contain following symbols:

- The apostrophe “ ’ ”
- The numbers 0 – 9
- The lower case character “j”
- The upper case characters “B”, “C”, “E”, “G”, “J”, “K”, “M”, “O”, “Q”, “R”, “V”, “X”, “Z”

CHAPTER 4

Evaluation

Different synthetic data sets containing HWT are introduced and discussed in Chapter 3. Those data sets are used as training data for task-specific deep learning models to evaluate to which extent they are meaningful for solving comparable tasks with real data. As already explained in Chapter 1, two domains are selected: handwritten text detection, and handwritten text recognition.

The problem of HTD is approached by training object detection models on the synthetic data and assessing their performance using the evaluation data sets presented in Chapter 3. The deep learning models YOLOv5 [JAS+] and YOLOv8 [JCQ] by Ultralytics are used for this purpose. HTR is approached by training the OCR network proposed by [CCP21] on synthetic HWT images. The trained models are evaluated on real HWT images to assess the appropriateness of synthetic data for HTR tasks. All models are trained on computers of the Vienna Scientific Cluster, more precisely the VSC-5, which offers two different GPU-enabled nodes containing either 2x NVIDIA A40 or 2x NVIDIA A10 GPUs. Both node types are used, depending on availability.

Section 4.1 discusses different object detection model configurations and the results when trained with different HTD data sets. Section 4.2 explains the modifications that had to be implemented for the VAN for HTR, as well as the results when trained on different HTR data sets.

4.1 Evaluating Synthetic HTD Data Sets

Section 4.1.1 gives an overview of the model configurations applied for the experiments and the metrics the results are assessed with. The trained models can be grouped into four groups. The first group, described in Section 4.1.2, focuses on assessing the effect of image color scales and HWT augmentation. Section 4.1.3 describes the result of different model architectures and sizes of the training data. The effect of using different labels (on paragraph-, line-, word-level) are described in Section 4.1.4. Enhancing the training data with fully synthetic images is assessed in Section 4.1.5. The model configuration and metrics are summarized in Tables 4.1, 4.2, 4.3, and 4.4 – one table for each of the four groups. Section 4.1.6 summarizes the findings.

4.1.1 Model Configuration and Metrics

The object detection networks YOLOv5 [JAS+] and YOLOv8 [JCQ] are trained on the training set of the synthetic HTD data sets described in Chapter 3. All models are trained to detect objects of one class only, namely areas containing HWT. Evaluation is done using the test set of the synthetic data, as well as the *CVL-** and *SCAN-** data sets presented in Chapter 3.

A standard metric to compare object detection models is the Mean Average Precision (mAP), which is the mean of all Average Precisions (AP) for all classes – since the HTD tasks have one class only, this is equal to the average precision of this class. The AP is the Area Under Curve (AUC) of the interpolated precision vs. recall curve for different prediction confidence levels. The mAP is often interpreted at a certain threshold of the Intersection Over Union (IoU), which indicates how much a predicted bounding box overlaps with a bounding box from the ground truth. The typical threshold of 0.5 is used in this work, meaning that the IoU must be at least 50%. The reported metric is therefore named mAP@50. [PPD+21] provides additional details about the AP and mAP.

The remaining metrics reported are based on pixel-level comparisons of the predictions vs. ground truth: a pixel either belongs to a bounding box containing HWT, or not; a pixel is either within a bounding box prediction, or not. This allows the definition of true positive, false positive, true negative, and false negative pixel predictions; this further enables the computation of pixel-level precision (P), recall (R), and F1 score (F1) for each image. Only predictions with a confidence level of at least 0.5 are considered. Therefore, the metrics P@50, R@50, and F1@50 are the averages of the pixel-level precision, recall, and F1 score of all images considering predictions with a confidence of at least 0.5.

Summarized, the following metrics are reported:

P@50 The average of the pixel-level precision for all images and predictions with a confidence level of at least 0.5.

R@50 The average of the pixel-level recall for all images and predictions with a confidence level of at least 0.5.

F1@50 The average of the pixel-level F1 score for all images and predictions with a confidence level of at least 0.5.

mAP@50 The mean average precision using an IoU threshold of 0.5 for predictions with a confidence level of at least 0.5.

Those metrics are reported for 16 different models, each trained with a different architecture, training data, or granularity of labels. As already mentioned in Chapter 2, the model architectures YOLOv5n (5n) [JAS$^+$], YOLOv8n (8n), and YOLOv8m (8m) [JCQ] are used. If not stated otherwise, models pre-trained on the COCO data set [LMB$^+$14] are used with the default parameters listed in Appendix D. All models are trained with the implementations' default optimizer, stochastic gradient descent with Nesterov momentum, with learning rate $\gamma = 0.1$ and momentum $\mu = 0.937$. Early stopping is enabled, with the default patience of 100 iterations for YOLOv5 models and 50 iterations for YOLOv8 models. The maximum epochs per training run was limited to 1000. The models are trained with the implementations' default hyperparameters (which differ between model architectures, see Appendix D), unless stated otherwise.

Predictions on data sets containing cutouts of size 640x640 pixels are merged to still obtain results comparable to full-sized data sets. The utilized algorithm can be explained using concepts known from graph theory: The predictions (i.e. bounding boxes) from the cutouts are mapped to their absolute position on the original image. Since the cutouts overlap, predictions will overlap as well. Each bounding box is interpreted as a node of a tree. Two nodes within a tree are considered as connected, if and only if their corresponding bounding boxes overlap. All predictions of a document together form one or multiple trees, hence a forest. The task of finding predictions to merge together is now equivalent to finding the individual trees within a forest. Once the trees are isolated, the final predictions are derived from the minima and maxima of the bounding box coordinates of all predictions within a tree.

4.1.2 Effect of Color Scales and Handwritten Text Augmentation

Table 4.1 lists results for models trained on two different kinds of data sets: *COLSCALES-*-CUT-PAR* with images in different color scales (RGB, grayscale and binarized images), and *COLSCALES-STROKE-BW-CUT-PAR* with binarized images and stroke width augmentation.

Images with RGB color scale provide the best results considering the synthetic data only: the mAP@50 is 0.86, the F1@50 is 0.89. However, using binarized training images decreases the mAP@50 for synthetic data to 0.82, but increases it from 0.27 to 0.42 for the *CVL* data, and from 0.17 to 0.52 for the *SCAN* data. The F1@50 is unchanged for the synthetic data, decreases slightly from 0.88 to 0.85 for the *CVL* data, but increases from 0.44 to 0.81 for the *SCAN* data. Introducing stroke width augmentation has no further effect for the synthetic data. However, the mAP@50 and F1@50 on the *CVL* data are increased from 0.42 to 0.47, and from 0.85 to 0.90, respectively; the mAP@50 for the *SCAN* data is increased from 0.52 to 0.56, but the F1@50 decreases from 0.81 to 0.69.

Using grayscale or binarized data yields strictly better results when evaluating on real data, likely due to less details the model has to filter out. Grayscale vs. binarized images do not have a clear outcome. However, using binarized images provides the best results for the *SCAN* data sets although stroke width augmentation has different effects on

	Model Description	P@50	R@50	F1@50	mAP@50
SYN	5n, 640, COLSCALES-RGB-CUT-PAR	**0.83**	**0.97**	**0.89**	**0.86**
	5n, 640, COLSCALES-GRAY-CUT-PAR	0.82	**0.97**	**0.89**	0.84
	5n, 640, COLSCALES-BW-CUT-PAR	**0.83**	0.96	**0.89**	0.82
	5n, 640, COLSCALES-STROKE-BW-CUT-PAR	0.82	0.96	**0.89**	0.82
CVL	5n, 640, COLSCALES-RGB-CUT-PAR	**0.91**	0.85	0.88	0.27
	5n, 640, COLSCALES-GRAY-CUT-PAR	0.90	0.88	0.89	0.35
	5n, 640, COLSCALES-BW-CUT-PAR	**0.91**	0.80	0.85	0.42
	5n, 640, COLSCALES-STROKE-BW-CUT-PAR	0.89	**0.90**	**0.90**	**0.47**
SCAN	5n, 640, COLSCALES-RGB-CUT-PAR	0.60	0.35	0.44	0.17
	5n, 640, COLSCALES-GRAY-CUT-PAR	0.76	**0.82**	0.79	0.55
	5n, 640, COLSCALES-BW-CUT-PAR	**0.83**	0.80	**0.81**	0.52
	5n, 640, COLSCALES-STROKE-BW-CUT-PAR	0.65	0.74	0.69	**0.56**

Table 4.1: Description and metrics for HTD models trained on different data sets. Each row within a segment refers to a model trained on the architecture, image size, and data set described in the second column. The models across segments are equal, but the test data differs, as indicated in the first column. The highest metrics are shown in bold font.

F1@50 and mAP@50, but image binarization together with stroke width augmentation provides the best results for the *CVL* data. Hence, all following experiements are done one binarized images with stroke width augmentation.

4.1.3 Assessing Different Model Architectures and Image Sizes

Considering different model architectures reveals that models based on the YOLOv8 architecture perform better than those based on the YOLOv5 architecture, as shown in Table 4.2. The best results on synthetic data are achieved using a YOLOv8m model with training data sized 1280x1280 pixels; the mAP@50 is 0.96, the F1@50 is 0.97.

The best model regarding the *CVL* data is a YOLOv8n model trained on cutouts of 640x640 pixels: the mAP@50 is 0.78, the F1@50 is 0.90. However, the mAP@50 for the other models is significantly lower, and has with 0.02 its minimum; but the best performing model on the synthetic data (YOLOv8m with image sizes pf 1280x1280 pixels) still yields a F1@50 of 0.85, whereas the mAP@50 is 0.38. The precision of 0.95 indicates that HWT was correctly identified in the sense that if the model predicted an HWT bounding box, it contained HWT to a large extent. But since the paragraphs for the *CVL* data are relatively big compared to the synthetic images, the predicted bounding boxes also need to be larger, otherwise the predictions would be too small. This would result in a low recall, a low F1, a low IoU, many false positives, and finally a low mAP@50.

	Model Description	P@50	R@50	F1@50	mAP@50
SYN	5n, 1280, FULLSIZE-STROKE-BW-PAR	0.95	0.93	0.94	0.95
	8n, 640, FULLSIZE-STROKE-BW-PAR	0.95	0.94	0.95	0.87
	8n, 1280, FULLSIZE-STROKE-BW-PAR	0.97	0.95	0.96	0.94
	8m, 1280, FULLSIZE-STROKE-BW-PAR	**0.98**	**0.97**	**0.97**	**0.96**
CVL	5n, 1280, FULLSIZE-STROKE-BW-PAR	0.94	0.14	0.24	0.02
	8n, 640, FULLSIZE-STROKE-BW-PAR	**0.96**	**0.84**	**0.90**	**0.78**
	8n, 1280, FULLSIZE-STROKE-BW-PAR	0.95	0.76	0.85	0.48
	8m, 1280, FULLSIZE-STROKE-BW-PAR	0.95	0.77	0.85	0.38
SCAN	5n, 1280, FULLSIZE-STROKE-BW-PAR	0.84	**0.80**	**0.82**	0.51
	8n, 640, FULLSIZE-STROKE-BW-PAR	0.87	0.67	0.75	**0.53**
	8n, 1280, FULLSIZE-STROKE-BW-PAR	**0.89**	0.76	**0.82**	0.44
	8m, 1280, FULLSIZE-STROKE-BW-PAR	**0.89**	0.69	0.78	0.41

Table 4.2: Description and metrics for HTD models trained using different model architectures and image sizes. Each row within a segment refers to a model trained on the architecture, image size, and data set described in the second column. The models across segments are equal, but the test data differs, as indicated in the first column. The highest metrics are shown in bold font.

Using cutouts mitigates this problem, as the distribution of bounding box sizes is more equal for synthetic and *CVL* data.

This problem exists for the *SCAN* data as well, but is less severe. Using cutouts results again in the highest mAP@50 of 0.53, but the lowest value is 0.26 using a YOLOv8n model. The highest F1@50 is 0.82, achieved on both the YOLOv5n and YOLOv8n models. Although the YOLOv5 architecture provides the best results for the *SCAN* data, using models based on the YOLOv8 architecture are used for the remaining experiments, as they excel for both the synthetic and the training data.

4.1.4 Effect of the Label Granularity

As mentioned in Section 4.1.3, the distribution of bounding box sizes of the training data should match that of the target data the model is trained for. Using synthetic HWT easily allows to reduce the label granularity from entire paragraphs to lines or single words, which is one method to align the bounding box sizes. Table 4.3 summarizes the results for models trained with equal data, but different labels.

Notably is, again, that using labels on paragraph level yields the lowest mAP@50 (0.13 for the *CVL* data, 0.30 for the *SCAN* data), while the precision is at 0.98 for the *CVL* data and 0.96 for the *SCAN* data. Reducing the label granularity increases both the

	Model Description	P@50	R@50	F1@50	mAP@50
SYN	8n, 1280, GRANULARITY-STROKE-BW-PAR	**0.97**	0.96	**0.97**	0.94
	8n, 1280, GRANULARITY-STROKE-BW-LINE	**0.97**	**0.97**	**0.97**	**0.97**
	8n, 1280, GRANULARITY-STROKE-BW-WORD	0.96	0.96	0.96	0.89
CVL	8n, 1280, GRANULARITY-STROKE-BW-PAR	**0.98**	0.76	0.85	0.13
	8n, 1280, GRANULARITY-STROKE-BW-LINE	0.94	**0.92**	**0.93**	**0.79**
	8n, 1280, GRANULARITY-STROKE-BW-WORD	0.88	0.86	0.87	0.76
SCAN	8n, 1280, GRANULARITY-STROKE-BW-PAR	**0.96**	0.74	0.84	0.30
	8n, 1280, GRANULARITY-STROKE-BW-LINE	**0.96**	**0.81**	**0.88**	0.56
	8n, 1280, GRANULARITY-STROKE-BW-WORD	0.95	0.69	0.80	**0.67**

Table 4.3: Description and metrics for HTD models trained for different label granularities. Each row within a segment refers to a model trained on the architecture, image size, and data set described in the second column. The models across segments are equal, but the test data differs, as indicated in the first column. The highest metrics are shown in bold font.

F1@50 and the mAP@50 for both real data sets: Using labels on line-level provides the best results for the synthetic data (mAP@50 and F1@50 are both 0.97) and for the *CVL* data (mAP@50 is 0.79, F1@50 is 0.93). Although the F1@50 using line-level labels is with 0.88 the highest for the *SCAN* data, the highest mAP@50 of 0.67 is achieved using labels on word-level.

4.1.5 Using Fully Synthetic Data

Another approach to align the bounding box distributions is to add bigger HWT paragraphs, as done with the *CWT-STROKE-BW-** data sets, which are equal to the *GRANULARITY-STROKE-BW-** data sets but with additional 1,000 fully synthetic images. The results for models trained on this data are listed in Table 4.4.

Notably is the mAP@50 for the *CVL* and *SCAN* data when trained on the YOLOv8 model compared with the results of the same model architecture shown in Table 4.3. Adding fully synthetic data with bigger paragraphs increased the mAP@50 from 0.13 to 0.84 for the *CVL* data, and from 0.30 to 0.37 for the *SCAN* data.

The main motivation for using this data set is to allow a more diverse placement of HWT. Magazines or papers tend to have most of the content centered in the page, with only limited space between paragraphs or columns. Hence, most of the HWT must be placed towards the borders of the image; using completely synthetic data, where CWT and HWT are randomly placed, allows to break up this structure.

	Model Description	P@50	R@50	F1@50	mAP@50
SYN	8n, 1280, CWT-STROKE-BW-PAR	0.97	0.97	0.97	0.94
	8n, 1280, CWT-STROKE-BW-LINE	**0.98**	0.97	0.97	**0.97**
	8n, 1280, CWT-STROKE-BW-WORD	0.95	0.97	0.96	0.90
	8m, 1280, CWT-STROKE-BW-PAR	**0.98**	**0.98**	**0.98**	0.96
	8m, 1280, CWT-STROKE-BW-LINE	**0.98**	**0.98**	**0.98**	**0.97**
CVL	8n, 1280, CWT-STROKE-BW-PAR	0.98	**0.93**	0.95	0.84
	8n, 1280, CWT-STROKE-BW-LINE	0.94	**0.93**	0.93	0.82
	8n, 1280, CWT-STROKE-BW-WORD	0.88	0.88	0.88	0.78
	8m, 1280, CWT-STROKE-BW-PAR	**0.99**	**0.93**	**0.96**	**0.88**
	8m, 1280, CWT-STROKE-BW-LINE	0.95	**0.93**	0.94	0.84
SCAN	8n, 1280, CWT-STROKE-BW-PAR	**0.98**	0.80	0.88	0.37
	8n, 1280, CWT-STROKE-BW-LINE	0.96	**0.83**	**0.89**	0.62
	8n, 1280, CWT-STROKE-BW-WORD	0.95	0.77	0.85	**0.72**
	8m, 1280, CWT-STROKE-BW-PAR	0.96	0.75	0.84	0.48
	8m, 1280, CWT-STROKE-BW-LINE	0.97	**0.83**	**0.89**	0.59

Table 4.4: Description and metrics for HTD models trained with data sets containing synthetic CWT. Each row within a segment refers to a model trained on the architecture, image size, and data set described in the second column. The models across segments are equal, but the test data differs, as indicated in the first column. The highest metrics are shown in bold font.

However, this approach yields little improvement using labels on line and word level compared to labels on paragraph level. For the synthetic data, the mAP@50 increased only for labels on word-level, from 0.89 (in Table 4.3) to 0.90. Using the *CVL* data set and labels on line-level, the mAP@50 increased from 0.79 to 0.82, the F1@50 is unchanged; using labels on word-level increases the mAP@50 from 0.76 to 0.78, the F1@50 from 0.87 to 0.88. A stronger effect is observed on the *SCAN* data: on line level, the mAP@50 is increased from 0.56 to 0.62, the F1@50 from 0.88 to 0.89; on word level, the mAP@50 is increased from 0.67 to 0.72, the F1@50 from 0.80 to 0.84.

Using YOLOv8m models instead of YOLOv8n further increases the performance when evaluated on synthetic or *CVL* data. The best results for those data sets are achieved with labels on line level: for the synthetic data, a mAP@50 of 0.97 and a F1@50 of 0.98 are reached; for the *CVL* data, a mAP@50 of 0.88 and a F1@50 of 0.96 is reached. For the *SCAN* data, the highest F1@50 of 0.89 is likewise achieved using a YOLOv8m model trained with labels on line-level. However, the highest mAP@50 of 0.72 is reached with a YOLOv8n model with labels on word level. No YOLOv8m model could be trained with this label granularity due to hardware limitations, hence it is possible that this architecture would outperform YOLOv8n as well.

4.1.6 Summary

Using full-size images and models based on the YOLOv8 architecture yields the best results. The distribution of label sizes should be similar between training and test data to achieve a high mAP@50. This is possible due to the use of synthetic data, as it allows to adjust the granularity of labels and the size of paragraphs in the training data. Overall, a mAP@50 of 0.88 and a F1@50 of 0.96 is reached for the *CVL* data set; and a mAP@50 of 0.72 and a F1@50 of 0.89 for the *SCAN* data set. Summarized, synthetic HWT allows generation of images tailored to the respective target data. Apart from selecting the correct model architecture (YOLOv8m for the *CVL* data, YOLOv8n or YOLOv8m for the *SCAN* data), the following had the biggest impact:

- The possibility of generating paragraphs of arbitrary size
- The possibility of mimicking different stroke widths
- The possibility of generating labels with different granularity
- The possibility to generate data sets of arbitrary size

4.2 Evaluating Synthetic HTR Data Sets

This section is divided into four parts: First, the HTR model and metrics are described in Section 4.2.1. Section 4.2.2 contains the baseline for the HTR model if trained on real data only, and describes the results if trained on synthetic data only and evaluated on real data – for both line and paragraph images. The properties of different synthetic and real data sets and their effect on the model performance are compared and discussed in Section 4.2.3. Results of mixing synthetic and real data for training are the content of Section 4.2.4.

4.2.1 Model and Metrics

As already explained in Chapter 2, the model proposed by [CCP21] is a VAN for end-to-end paragraph transcription with implicit line segmentation. Hence, images containing entire paragraphs or single lines only can be used as training data. The source code provided by the authors is used as-is except following changes potentially affecting the model performance:

- The usage of NVIDIA Apex[1], a package for mixed precision and distributed training, is replaced with the features for automatic mixed precision and distributed training

[1] `https://nvidia.github.io/apex/`, last accessed on 2023-08-19

built-in into PyTorch[2], the Python package the HTR model is built with. The reason is software incompatibilities.

- Image binarization is added as an optional pre-processing step as an attempt to improve prediction performance.

The image pre-processing and data augmentation steps for the HTR models described in this thesis are the same as described in the original paper. The only difference is image normalization and binarization, which are two pre-processing steps that are additionally done for some models (the respective pre-processing steps applied for each model are explained in Section 4.2.2). The results reported in the original paper are trained on gray-scale images, which is done as well within this thesis (except for models trained on binarized images). The Adam optimizer with a learning rate of 10^{-4} is used. The original models were trained on one GPU with a training time limited to two days; the best model found in this time period is reported. The batch size is set to 16 for line-, and 8 for paragraph training. If not stated otherwise, the models reported in this thesis are trained on two GPUs, the training time was limited to one day or 3000 epochs, whatever is reached first; the best model based on the evaluation split of the data (or the test data set if no evaluation data is available, as for the CVL data set [KFDS13]) within this period is reported. All line-based models are trained with a batch size of 32, and with a batch size of 8 for all paragraph models; the values for the original models are 16 and 8 for line-based and paragraph-based models, respectively.

Two metrics are reported in the original paper [CCP21]: The Character Error Rate (CER), and the Word Error Rate (WER). The CER and WER allow for assessing how many characters or words, respectively, have been wrongly transcribed by the OCR model. The computation for both is similar; the general formula for the error rate is shown in Equation 4.1[3]. This formula should be interpreted as follows for the CER: S is the number of characters that were substituted in the transcription, D is the number of characters that were deleted, I is the number of characters that were additionally inserted, C is the number of characters which are correctly transcribed, N is the total number of actual characters, hence the sum of all substitutions, deletions and correct transcription, hence $N = S + D + C$. The same holds for the WER, but with the following interpretation: S is the number of substituted words, D is the number of deleted words, and so forth.

$$ER = \frac{S + D + I}{S + D + C} \quad = \frac{S + D + I}{N} \tag{4.1}$$

Notably is that the values for both the CER and WER do not have an upper limit of 1: If many characters or words are additionally inserted for the transcription so that I

[2] https://pytorch.org/, last accessed on 2023-08-19

[3] Taken from the PyTorch documentation, https://torchmetrics.readthedocs.io/en/stable/text/char_error_rate.html, last accessed on 2023-09-08

		IAM		CVL	
Training Data	PP	CER	WER	CER	WER
IAM	–	**5.2**	**17.2**	12.2	43.0
IAM	N	5.6	18.0	**9.9**	**30.7**
IAM	N, B	5.8	18.6	13.3	46.3
CVL	–	–	–	3.7	11.2
CVL	N	–	–	3.8	12.0
CVL	N, B	–	–	**3.3**	**10.4**

Table 4.5: Evaluation results of HTR models trained on real HWT line images. The first column contains the training data, the second one the pre-processing steps (PP) – image normalization (N), image binarization (B), or none of them. The third and fourth column contain the CER and WER for the IAM data, the last two columns for the CVL data. The same pre-processing steps are applied for training and test data. The best results for each section are presented in bold font. All values are in percent.

is large enough so that $S + D + I > N$ holds, then the error rate exceeds 1. Further, the WER is naturally larger than the CER because words are composed of multiple characters; hence the denominator of the error rates, N, is smaller. Finally, both metrics can be interpreted as a percentage of wrongly predicted characters/words.

4.2.2 Training a Handwriting Recognition Model on Synthetic Data

Baseline for HWT line images The baseline reported in [CCP21] for a model trained and evaluated on the IAM data set is a CER of 5.1% and a WER of 16.5%. Those values could not be reproduced precisely, but a CER of 5.2% and a WER of 17.2% are reached, as shown in Table 4.5. Data normalization or using binarized images does not improve the model performance on that data.

Training and evaluating the model on images taken from the CVL data set yields lower error rates than for the IAM data set. Without any pre-processing steps, a CER of 3.7% and a WER of 11.2% is achieved. Using data normalization and image binarization provides the best results, the CER is 3.3%, the WER is 10.4%. An explanation is that the CVL images have a lower contrast compared to the IAM images (the average Michelson contrast is 1.85 for the CVL data and 6.7 for the IAM data; the average root mean square contrast is 25.62 for the CVL data and 43.64 for the IAM data), since binarization increases the contrast, it has more effect on the CVL data; a higher contrast means that the text is more distinct from the background, which might help for OCR.

Applying inter-data set evaluation, i.e., training on images from the IAM data set and evaluating the model on the CVL data, yields a CER of 9.9% and a WER of 30.7%, as shown in Table 4.5. Training on synthetic data and evaluating on real data is an inter-data set evaluation as well. Since the best results using real data only are achieved with image normalization enabled, this pre-processing step is used for further experiments.

		SYN		IAM		CVL	
Training Data	PP	CER	WER	CER	WER	CER	WER
LOTR-LINE	–	5.3	10.4	**28.3**	**65.5**	**30.0**	**74.0**
GUT-LINE, N	N	4.6	10.5	29.8	66.7	33.3	75.8
LOB-LINE, N	N	0.6	2.9	33.3	70.5	36.8	91.2
LOB-50K-LINE, N	N	0.4	2.0	30.8	66.7	32.6	75.4
LOB-SEED-LINE, N	N	0.5	1.9	32.4	70.0	34.3	78.7
LOB-UNIFORM-LINE, N	N	0.7	3.3	32.8	71.0	34.6	79.2
LOB-RAW-LINE, N	N	26.0	43.8	71.8	101.0	70.9	104.7
LOB-NOAUG-LINE, N	N	**0.2**	**1.2**	29.0	66.3	34.0	87.5
LOB-LINE, N / B	N, B	0.7	3.1	31.6	68.8	35.7	79.8

Table 4.6: Evaluation results of HTR models trained on HWT line images. The first column contains the training data, the second one additional pre-processing steps (PP) – image normalization (N), image binarization (B), or none of them. The third and fourth column contain the CER and WER for the test split of the synthetic data, the fifth and sixth column for the IAM data, the last two columns for the CVL data. The same pre-processing steps are applied for training and test data. The best results are presented in bold font. All values are in percent.

Training on synthetic line images Table 4.6 shows the results for training on synthetic data only. All models presented in this table are trained on different data sets or image pre-processing steps. Intra-data set generalization works, in general, quite well. The lowest error rates are a CER of 0.2% and a WER of 1.2%, achieved when trained on the *LOB-NOAUG-LINE* data set with normalization enabled.

However, inter-data set generalization does not reach the state-of-the-art performance of the baseline models; the error rates are more than tripled: the CER and WER on the IAM data are 28.3% and 65.5%, respectively; the CER and WER for the CVL data is 30.0% and 74.0%, respectively. Those values are all achieved with the model trained on the synthetic *LOTR-LINE* data set.

Those results are robust against the usage of different text corpora. Using the *GUT* text corpus with the *GUT-LINE* data set does not improve the error rates (CER and WER are 29.8% and 66.7%, respectively, for the IAM data, and 33.3% and 75.8%, respectiely, for the CVL data). The same holds for using the *LOB* text corpus on which the *LOB-LINE* and IAM data sets are based on (CER and WER are 33.3% and 70.5% for the IAM data, and 36.8% and 91.2% for the CVL data).

Neither increasing the training data from 10k images to 50k images (as done with *LOB-50K-LINE*, the maximum training time is doubled from one to two days for this data set), nor using a different seed for the image generation model (as done with *LOB-SEED-LINE*), nor using a different distribution for the style vectors for the image generation model (as done with *LOB-UNIFORM-LINE*) improves the error rates. Using

		IAM		CVL	
Training Data	PP	CER	WER	CER	WER
IAM-LINE / IAM-PAR	–	4.8	16.4	13.0	37.2
CVL-LINE / CVL-PAR	N	–	–	5.0	15.9

Table 4.7: Evaluation results of HTR models trained on real HWT paragraph images. The first column contains the training data (the VAN is pre-trained on line images), the second one contains pre-processing steps (PP) – normalized (N) vs. no pre-processing. The third and fourth column contain the CER and WER for the IAM data, the last two columns for the CVL data. The same pre-processing steps are applied for line and paragraph data, and for training and test sets. All values are in percent.

no image augmentation at all (as done with *LOB-RAW-LINE*) or only image resizing but no stroke augmentations (as done with *LOB-NOAUG-LINE*), or image binarization as pre-processing step, do not improve the inter-data set error rates either.

In summary, synthetic data alone is insufficient to get comparable results on HTR tasks for line images. However, a closer look at the properties of the synthetic data sets reveals that the distribution of image sizes and pixels per character show significant differences between synthetic and real HWT data sets, hence affecting the model performance negatively. A detailed discussion can be found in Section 4.2.3. In general, the following factors likely influenced the results:

- The applied model architecture does, in general, perform poorly on inter-data set generalization. The error rates for the CVL images of a model trained on IAM data are about three times as high as if trained directly on the CVL data (see Table 4.5).
- The generated HWT images do sometimes have unrealistic styles, as already discussed in Chapter 2.
- Some properties of the generated HWT images (e.g., pixels/character) show major differences compared to the real data sets (see Section 4.2.3).

Baseline for HWT paragraph images The applied HTR model architecture uses a two-step approach for training on paragraph-level. The first one consists of pre-training parts of the VAN on line images. The second one is training on paragraph images using the pre-trained weights. The baseline models following this approach, i.e. pre-training on real HWT line images followed by real paragraph images, are listed in Table 4.7.

A CER of 4.8% and a WER of 16.4% are achieved for the IAM data set, which is quite close to the values mentioned in the original paper from [CCP21], where a CER of 4.45% and a WER of 14.55% is reported. The results for the CVL data set are similar: a

		SYN		IAM		CVL	
Training Data	PP	CER	WER	CER	WER	CER	WER
GUT-LINE / GUT-PAR	N	5.9	12.8	33.3	71.9	32.9	76.4

Table 4.8: Evaluation results of HTR models trained on HWT paragraph images. The first column contains the training data (the VAN is pre-trained on line images), the second one the applied pre-processing steps (PP) – only normalization (N) is applied. The third and fourth column contain the CER and WER for the test split of the synthetic data, the fifth and sixth column for the IAM data, the last two columns for the CVL data. The same pre-processing steps are applied for line and paragraph data, and for training and test sets. All values are in percent.

CER of 5.00% and a WER of 15.87% is achieved, but only by normalizing the line and paragraph images (as opposed to not normalizing as done for the IAM data).

Training on synthetic paragraph images Pre-training using synthetic data yields similar results as using line data only, as shown in Table 4.8: evaluating the model on the synthetic test data yields a CER of 5.9% and a WER of 12.8%, but predictions on the IAM data yield a CER of 33.3% and a WER of 71.9%, and a CER of 32.9% and WER of 76.4% on the CVL images. The reason is probably the same as for the line data since the actual character recognition backbone applied to segmented lines is the same for line and paragraph recognition, and because the model detected the correct number of lines in 97.5% (2843 out of 2915) of all evaluation images. Hence, no further experiments regarding paragraph-level HTR are done.

4.2.3 Comparison of Synthetic and Real Data Sets for Handwritten Text Recognition

The synthetic data sets have properties with different distributions compared to the real data sets. Figure 4.1 shows the distribution of pixels per character for five different data sets: *IAM-LINE*, *CVL-LINE*, *LOB-RAW-LINE* (a synthetic data set without any augmentations), *LOTR-LINE* (a synthetic data set based on the *LOTR* text corpus with image augmentation) and *LOB-LINE* (a synthetic data set based on the *LOB* text corpus with image augmentation). The first two only contain HWT written by humans. Notably, their shape, location, and variance seem to be quite similar: most images have between 20 and 60 pixels/character with a peak at around 40 pixels/character, and a long right tail to around 140 pixels/character.

Another data set in the chart, *LOB-RAW-LINE*, contains synthetic images without any augmentations applied, hence also without image resizing. Notable is that most images have up to 40 pixels/character, with a peak at around 14 pixels/character, and again a long right tail. The different distributions of text length/image across the data sets, shown in Figure 4.2, is a partial explanation only: the number of characters on the images

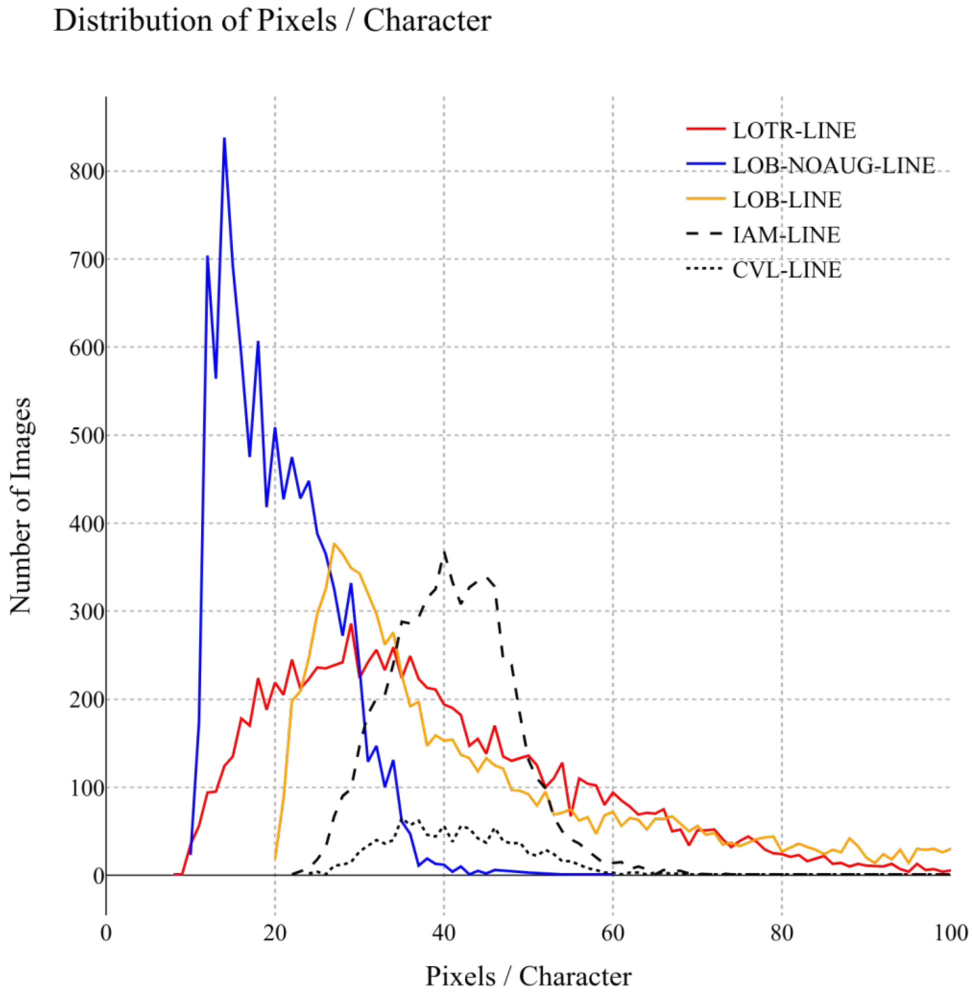


Figure 4.1: The distribution of pixels per character – computed by dividing the image width by the text length – for selected HTR data sets. *IAM-LINE* and *CVL-LINE* – the black lines – contain images written by humans. Notable is their matching distributions. The blue line represents the *LOB-RAW-LINE* data set which contains raw data only, i.e., without any augmentations or scaling applied. The other lines – red and orange – show data sets with different image scaling strategies. This image is zoomed in on the x-axis – the IAM and CVL data sets have a right tail until about 140 pixels/character, *LOTR-LINE* (red) until about 130 pixels/character, and *LOB-LINE* (orange) until about 600 pixels/character.

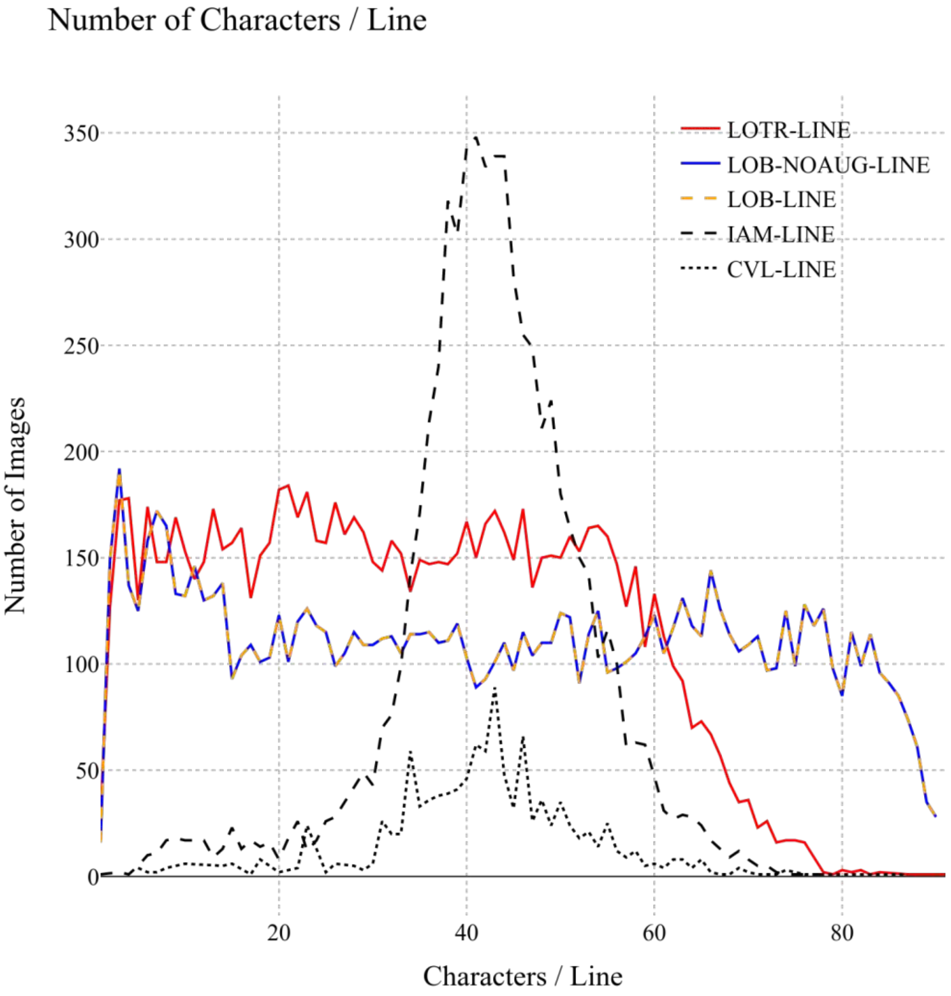


Figure 4.2: The distribution of characters/line for selected HTR data sets. *IAM-LINE* and *CVL-LINE* – the black lines – contain images written by humans. Notable are their matching distributions. The data set *LOTR-LINE* (red) has at most 12 words, and the data sets *LOB-RAW-LINE* (blue) and *LOB-LINE* (orange) have at most 90 characters.

follows more a uniform distribution for synthetic data and more a normal distribution for real data; however, the distribution of pixels/character for the synthetic data is positively skewed, and the maximum extent is at fewer pixels/character than for the real data. Hence, although trained on the IAM data set, the GAN generating the synthetic HWT images tends to create more narrow fonts than the training data.

This property is affected by image scaling: Figure 4.1 shows the pixels/character distribution for the *LOTR-LINE* data set also, which contains images that were randomly scaled by 80% – 300%. The peak of the distribution was shifted to 29 pixels/characters, but it

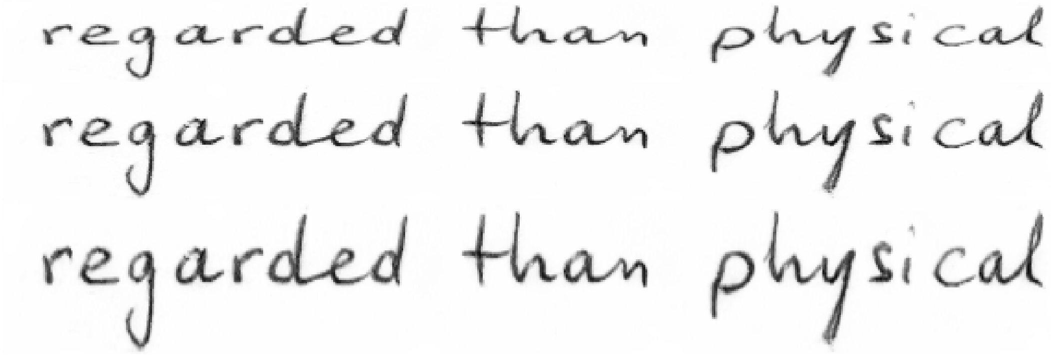


Figure 4.3: All three lines show the text "regarded than physical", which is a random excerpt from the *LOB* corpus. The middle line was created using the HWT generation model proposed by [DMP+20]. The top line has the same width as the middle line but was resized to have a height that is 30% smaller. Similar to the bottom line, which has a 30% bigger height than the middle line. Notable is that resizing HWT images without keeping the aspect ratio can still generate realistic images.

also extended the right tail from 60 pixels/character for the *LOB-RAW-LINE* data set to 130 pixels/character. Hence, random scaling alone is insufficient to obtain a similar distribution to the real images.

The HWT synthesis model proposed by [DMP+20] only allows an indirect influence on the font width via the style vector. Hence, applying a different image scaling strategy to the synthetic raw images is an attempt to arrange the pixels/character distribution better between real and synthetic data sets. The main idea is to use the image widths as proxy for the pixels/character by aligning the distributions of the widths. Instead of randomly scaling the images, the target scale is computed by standardizing the widths of the synthetic images, which are then shifted towards the distribution of the IAM data set. The new values are clipped to the upper limit of image widths of the IAM data set – around 2400 pixels – to avoid long right tails. This approach is formalized in Equation 4.2, where μ_{SYN}, σ_{SYN}, μ_{IAM} and σ_{IAM} are the mean and standard deviations of the widths of the synthetic images and the images from the IAM dataset, respectively, w is the width of the raw synthetic image, and w' is the width after scaling:

$$w' = min\left(\frac{w - \mu_{SYN}}{\sigma_{SYN}}\sigma_{IAM} + \mu_{IAM}, 2400\right) \tag{4.2}$$

The scaling factor for the entire image is computed using the new width w'. The images are not scaled with the aspect ratio to improve the diversity of the input images. Instead, the scaling factor for the image height is randomly distorted by ±30% from the scaling factor for the image width. The effect of not keeping the aspect ratio when scaling the HWT images is shown in Figure4.3.

This scaling strategy is applied for all data sets using the *LOB* corpus, except *LOB-RAW-LINE*. The effect of different scaling strategies on the image widths is shown in

Figure 4.4 using the *LOB-LINE* data set - applying the new approach shows that the width distributions of the *LOB-LINE* images better matches the distributions of the real data sets compared to random or no scaling: The mean of the image width went from 838 / 1249 pixels for *LOB-RAW-LINE* / *LOTR-LINE*, respectively, to 1614 pixels for *LOB-LINE*, while *IAM-LINE* and *CVL-LINE* have a mean of 1696 / 1686 pixels, respectively. Further, as shown in Figure 4.1, this strategy helps to eliminate quite narrow font widths - after scaling, the pixels/character start at around 20, as for the IAM data set. However, the peak of the distribution is again at around 29 pixels/character, and the distribution is again positively skewed

4.2.4 Training a Handwriting Recognition Model on Real and Synthetic Data

In the case where only small data sets with handwritten data are available, the combination with synthetic data brings a clear advantage for HTR (no more data has to be annotated manually for better results). Table 4.9 shows the results for experiments with the IAM data set [MB02], Table 4.10 for the CVL data set [KFDS13]. For mixed training, the models are pre-trained for 1500 epochs on the *LOB-LINE* data set followed by another 1500 epochs with the corresponding share of the real data. The models based on real data only are trained for 3000 epochs.

Using 10% of the IAM data set (which are 648 training images) results in a CER of 12.5% and a WER of 36.3%. Pre-training the model using 10,000 synthetic images reduces the CER by 3.3% to 9.2% and the WER by 8.5% to 27.8%. This effect decreases with an increasing number of real training data, and vanishes when using about 50% or more of the IAM training data, or 3,241 images. Figure 4.5a visualizes the results.

The results are more extreme on the CVL data set: using 10% of the training data only (or 113 images) results in a CER of 54.4% and a WER of 88.8%. Pre-training the model with 10,000 synthetic images reduces the CER by almost 40% to 14.6% and the WER by more than 45% to 43.4%. Notable is that the error rates of synthetic + real data and real data do not converge, as it is the case when using IAM images. Using about 45% of the CVL data, or 508 images, yields almost the same CER for both cases (7.9% for synthetic + real, 8.2% for real data only). The WER, however, is already 1.6% lower if real data only is used (26.2% vs. 24.6%). This effect increases with an increasing number of real training images. Overfitting is likely the reason for this: the CVL training data contains only five different texts, therefore similar words, or word sequences, occur with more varying styles compared to the IAM data or the synthetic data – from 1130 total training images, only 391 have a unique ground truth; for all of the remaining 739 images, at least one image shows exactly the same text. Adding synthetic images increases the training data's diversity, thus reducing the effect of overfitting. Figure 4.5b visualizes the results for the CVL data.

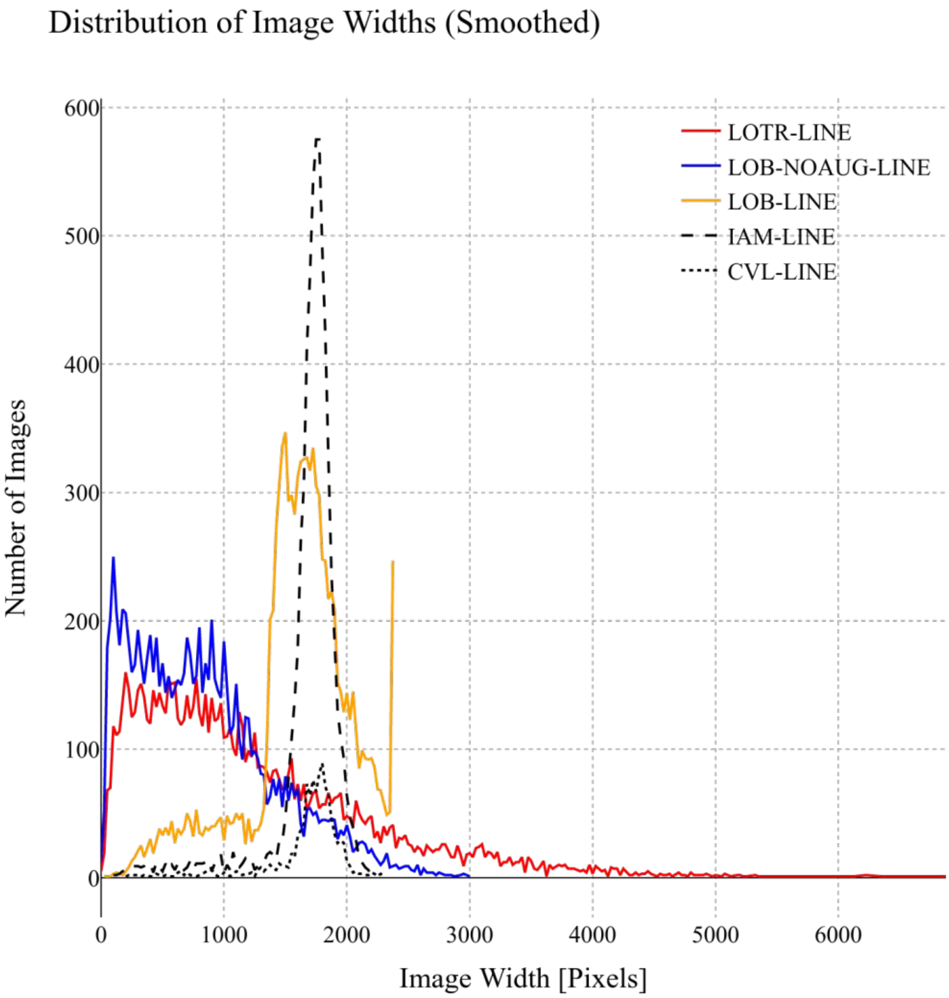


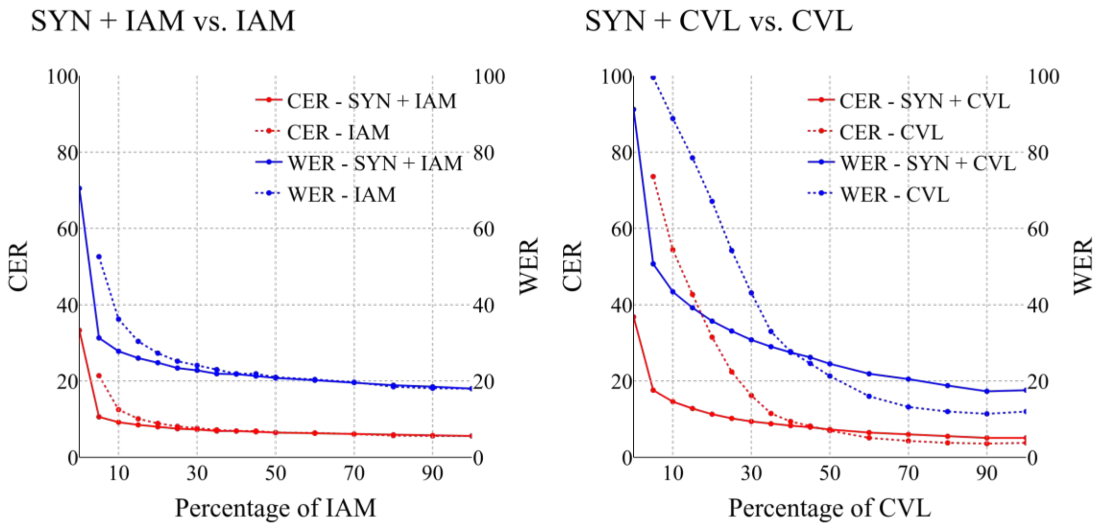
Figure 4.4: The distribution of image widths for selected HTR data sets. *IAM-LINE* and *CVL-LINE* – the black lines – contain images written by humans. Notable are their matching distributions. The blue line represents the *LOB-RAW-LINE* data set which contains raw data only, i.e., without any augmentations or scaling applied. Its width distribution is positively skewed; the peak is around 14 pixels. The red line represents *LOTR-LINE* containing randomly scaled images. Although random resizing helped to move the peak of the distribution a bit, it created a quite long right tail. The scaling strategy discussed in this section is applied for *LOB-LINE* (orange) - the distribution matches more closely the width distribution of the real images.

		SYN + Real		Real		Improvement SYN + Real vs. Real only		Δ Best Real vs. SYN + Real	
		CER	WER	CER	WER	CER	WER	CER	WER
0%	0	33.3	70.5	–	–	–	–	-27.7	-52.5
5%	324	10.6	31.3	21.4	52.6	10.8	21.3	-5.0	-13.3
10%	648	9.2	27.8	12.5	36.2	3.3	8.4	-3.6	-9.8
15%	972	8.5	26.0	10.1	30.4	1.6	4.4	-2.9	-8.0
20%	1,296	8.0	24.8	8.9	27.3	0.9	2.5	-2.4	-6.8
25%	1,620	7.5	23.4	8.1	25.2	0.6	1.8	-1.9	-5.4
30%	1,945	7.3	22.8	7.6	24.1	0.3	1.3	-1.7	-4.8
35%	2,269	6.9	21.9	7.2	23.0	0.3	1.1	-1.3	-3.9
40%	2,593	6.9	21.8	6.9	21.9	0.0	0.1	-1.3	-3.8
45%	2,917	6.7	21.3	6.9	21.9	0.2	0.6	-1.1	-3.3
50%	3,241	6.5	20.8	6.5	21.0	0.0	0.2	-0.9	-2.8
60%	3,889	6.3	20.2	6.4	20.4	0.1	0.2	-0.7	-2.2
70%	4,537	6.1	19.6	6.1	19.7	0.0	0.1	-0.5	-1.6
80%	5,186	5.9	18.9	5.7	18.5	-0.2	-0.4	-0.3	-0.9
90%	5,834	5.7	18.5	5.6	18.1	-0.1	-0.4	-0.1	-0.5
100%	6,482	5.6	18.0	5.6	18.0	0.0	0.0	0.0	0.0

Table 4.9: Evaluation results of HTR models trained on IAM or synthetic + IAM line images with different fractions of the real training data. The first two columns show the percentage and absolute value of real images. "SYN + Real" shows the results if the model is pre-trained on synthetic data for 1500 epochs, followed by another 1500 epochs with the respective fraction of real data. "Real" shows the results if trained with real data only for 3000 epochs. The improvement of using synthetic + real data over using real data only is listed in columns seven and eight – negative values indicate that using real data only yields better results, positive values indicate that using mixed data yields better results. The last two columns show the difference between the best result obtained with real data only (CER 5.6%, WER 18.0%) and "SYN + Real". The utilized synthetic data set is *LOB-LINE*. All results are in percent.

		SYN + Real		Real		Improvement SYN + Real vs. Real only		Δ Best Real vs. SYN + Real	
		CER	WER	CER	WER	CER	WER	CER	WER
0%	0	36.8	91.2	–	–	–	–	-33.0	-79.2
5%	56	17.6	50.7	73.6	99.6	56.0	48.9	-13.8	-38.7
10%	113	14.6	43.4	54.4	88.8	39.8	45.4	-10.8	-31.4
15%	170	12.8	39.2	42.7	78.5	29.9	39.3	-9.0	-27.2
20%	226	11.3	35.7	31.5	67.1	20.2	31.4	-7.5	-23.7
25%	282	10.2	33.1	22.4	54.2	12.2	21.1	-6.4	-21.1
30%	339	9.4	30.8	16.2	43.1	6.8	12.3	-5.6	-18.8
35%	396	8.8	29.0	11.5	33.0	2.7	4.0	-5.0	-17.0
40%	452	8.3	27.5	9.4	27.7	1.1	0.2	-4.5	-15.5
45%	508	7.9	26.2	8.2	24.6	0.3	-1.6	-4.1	-14.2
50%	565	7.3	24.5	7.0	21.3	-0.3	-3.2	-3.5	-12.5
60%	678	6.5	21.9	5.1	16.0	-1.4	-5.9	-2.7	-9.9
70%	791	6.0	20.5	4.3	13.2	-1.7	-7.3	-2.2	-8.5
80%	904	5.5	18.8	3.8	12.0	-1.7	-6.8	-1.7	-6.8
90%	1,017	5.1	17.3	3.6	11.4	-1.5	-5.9	-1.3	-5.3
100%	1,130	5.1	17.6	3.8	12.0	-1.3	-5.6	-1.3	-5.6

Table 4.10: Evaluation results of HTR models trained on CVL or synthetic + CVL line images with different fractions of the real training data. The first two columns show the percentage and absolute value of real images. "SYN + Real" shows the results if the model is pre-trained on synthetic data for 1500 epochs, followed by another 1500 epochs with the respective fraction of real data. "Real" shows the results if trained with real data only for 3000 epochs. The improvement of using synthetic + real data over using real data only is listed in columns seven and eight – negative values indicate that using real data only yields better results, positive values indicate that using mixed data yields better results. The last two columns show the difference between the best result obtained with real data only (CER 3.8%, WER 12.0%) and "SYN + Real". The utilized synthetic data set is *LOB-LINE*. All results are in percent.



(a)

(b)

Figure 4.5: The CER (red, left y-axis) and WER (blue, right y-axis) for models based on a varying amount of images (x-axis) from the IAM data set (a) or the CVL data set (b). The dotted lines represent models trained on real data only; the solid lines represent models trained on synthetic and real data.

CHAPTER 5

Conclusion

The potential of using synthetic data sets is examined for two domains: handwritten text detection and handwritten text recognition. On a high level, the approach is the same for both cases: synthetic data is generated, deep learning models solving the task at hand are trained on those data sets, and the applicability of the generated data sets is evaluated by applying the model on real data sets.

For HTD, synthetic HWT is added to scans of documents to mimic human annotations. Object detection models (YOLOv5 [JAS+] and YOLOv8 [JCQ]) are trained on this data to distinguish HWT from remaining content in the documents. The suitability of using synthetic data is evaluated by assessing the performance of those models on two data sets containing real HWT: the *CVL* data set [KFDS13], and a new *SCAN* data set explicitly created for this thesis, where handwritten text is manually added to a scientific paper.

The models are trained with different granularity of the labels: paragraphs, lines, and single words. The best models on the *CVL* data set achieved a mAP@50 of 0.88 and a F1@50 of 0.96 on paragraph level, a mAP@50 of 0.84 and F1@50 of 0.94 on line level, and a mAP@50 of 0.78 and F1@50 of 0.88 on word level; For the *SCAN* data set, a mAP@50 of 0.48 and a F1@50 of 0.84 are achieved on paragraph level, a mAP@50 of 0.62 and F1@50 of 0.89 on line level, and a mAP@50 of 0.72 and F1@50 of 0.85 on word level.

Summarized, synthetic HWT allows the generation of images tailored to the target data. Considering the results, following had the most significant impact:

- Generating paragraphs of arbitrary size and the possibility to mimic different stroke widths, as this allows to align the properties of the synthetic data with those of the target data
- The possibility of generating labels with different granularity allows to account for imperfections inherent to the data (e.g., varying sizes of white spaces or line lengths)

- It is easily possible to generate data sets of arbitrary size to adjust for model sizes

The research question RQ1, “To which extent can synthetic handwritten text be used to improve handwritten text detection models for annotated documents?” is therefore answered as follows: Using synthetic data allows control over the properties of the training data, which would not be possible with real data only. This, together with selecting the correct model architecture, allows to achieve a F1@50 of up to 0.96, and a mAP@50 of up to 0.88 on real data.

The domain of HTR is examined by generating synthetic images containing HWT, both lines and entire paragraphs. The HTR model proposed by [CCP21] is trained on those images and evaluated on the IAM [MB02] and CVL [KFDS13] data sets. With this approach a CER of 28.3% and a WER of 65.5% can be achieved for IAM line images (training on real data only yields a CER of 5.2% and a WER of 17.2% for IAM line data), and a CER of 30.0% and a WER of 74.0% on the CVL data set; the training data for the best models is based on the *LOTR* text corpus (see Chapter 3). Neither increasing the number of training images, using different text corpora, utilizing other image augmentation methods or variances in the synthetic image generation process led to better results. A CER of 33.3% and a WER of 71.9% are achieved on the IAM data set on paragraph level (training on real data only yields a CER of 4.8% and a WER of 16.4%); a CER of 32.9% and a WER of 76.4% are reached for the CVL data. The training data for those models is based on the *GUT* text corpus; synthetically generated images are vertically stacked to form paragraphs, including stroke width and stroke color augmentation, and line rotations before stacking.

While using synthetic data alone to train HTR models does not yield competitive performance, combining synthetic and real data does have advantages for small data sets containing line images. Using 10% of the IAM training data on line-level, or 648 images, achieves a CER and WER of 12.5% and 36.2%, respectively. Pre-training the model on 10,000 synthetic images reduces the error rates to 9.2% and 27.8%, respectively. This holds with an increasing number of real training data until about 50%, or 3241 images – although with decreasing improvement. Applying the same strategy to the CVL data yields a CER and WER of 54.4% and 88.8%, respectively, if the model is trained on 10% of the training data only, or 113 images. Pre-training the model with synthetic images reduces the CER to 14.6% and the WER to 43.4%. However, training on real images alone achieves better results if the model is trained with 45% (508 images) of the data or more, which is likely due to overfitting as the CVL data shows a smaller variety of content compared to the IAM images.

Those results do not contradict the findings reported by Fogel et al. [FAEC^{+}20]: training their HTR model on word level using all training images from the IAM database together with 100,000 synthetic images improved the WER from 12.24% to 11.68%, and the CER from 3.81% to 3.57%. Similar results are found within this thesis, but the improvement of the error rates is only observed if 50% or less of the real data is used. This might have two possible reasons: First, the model used within this thesis [CCP21] shows better error rates even if no data augmentation is done; hence, there's less room for improvement. Secondly, as discussed in Section 4.2.3, the images of the HWT generation model used within this thesis [DMP+20] show different statistics compared to the real data, especially regarding the distribution of pixels per character. The HWT generation method developed by Fogel et al. has a fixed width per character (32 pixels if normalized to an image height of 64 pixels), which more closely matches the peak of the actual distribution (around 40 pixels).

Summarized, the research question RQ2, "To which extent can synthetic handwritten text be used to improve handwritten text recognition models on line- and paragraph-level?" is answered as follows: Using synthetic training data alone does not yield competitive performance compared to state of the art models. The differences in the statistics of the data, especially the distribution of pixels/character, might be an important reason. However, mixing synthetic and real data can significantly improve the error rates, especially for small data sets containing only a few hundred real images. This is not only beneficial for existing ones, but also for data sets to-be created as it helps to save time while labelling raw data.

Aligning the properties of synthetic images with those of real HWT data (e.g. pixels/character) seems to be a promising approach to improve the HTR performance. Re-creating the results with another HWT generation model (e.g. [FAEC+20] – due to its fixed character width – or [BKC+21] – due to its better writer adaption compared to [DMP+20]) to observe the effect of different synthetic HWT data on the HTR performance would give a first outlook whether such an undertaking can be successful. Adapting the HWT generation models to historic handwriting to generate synthetic training data could help to improve the digitization of documents within this domain by reducing the required annotation effort.

Bibliography

[ABPP09] Antonacopoulos Apostolos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. A Realistic Dataset for Performance Evaluation of Document Layout Analysis. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR2009)*, pages 296–300, 2009.

[AYX+23] Ershat Arkin, Nurbiya Yadikar, Xuebin Xu, Alimjan Aysa, and Kurban Ubul. A survey: object detection methods from CNN to transformer. *Multimedia Tools and Applications*, 82(14):21353–21383, 2023.

[BKC+21] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. Handwriting Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1086–1094, October 2021.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc., 2009.

[BKL+19] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4715–4723, 2019.

[CCP21] Denis Coquenet, Clément Chatelain, and Thierry Paquet. SPAN: A Simple Predict & Align Network for Handwritten Paragraph Recognition. In *Document Analysis and Recognition – ICDAR 2021*, pages 70–84, Cham, 2021. Springer International Publishing.

[CCP22] Denis Coquenet, Clement Chatelain, and Thierry Paquet. End-to-End Handwritten Paragraph Text Recognition Using a Vertical Attention Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):508–524, 2022.

[CMV+19] Manuel Carbonell, Joan Mas, Mauricio Villegas, Alicia Fornés, and Josep Lladós. End-to-End Handwritten Text Detection and Transcription in Full

Pages. In *2019 International conference on document analysis and recognition workshops (ICDARW)*, volume 5, pages 29–34. IEEE, 2019.

[Coq22] Denis Coquenet. *Towards End-to-end Handwritten Document Recognition*. PhD thesis, Rouen University, France, 2022.

[DMP+20] Brian L. Davis, Bryan S. Morse, Brian L. Price, Chris Tensmeyer, Curtis Wigington, and Rajiv Jain. Text and Style Conditioned GAN for Generation of Offline Handwriting Lines. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020.

[FAEC+20] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2020.

[FS15] Stefan Fiel and Robert Sablatnig. Writer Identification and Retrieval Using a Convolutional Neural Network. In *Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015, Proceedings, Part II 16*, pages 26–37. Springer, 2015.

[GCBG09] Emmanuèle Grosicki, Matthieu Carré, Jean-Marie Brodin, and Edouard Geoffrois. Results of the RIMES Evaluation Campaign for Handwritten Mail Processing. In *2009 10th International Conference on Document Analysis and Recognition*, pages 941–945, 2009.

[Gra13] Alex Graves. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850, 2013.

[JAS+] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, Imyhxy, Lorna, Zeng Yifu, Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, Tkianai, YxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. 2022. URL: `https://zenodo.org/record/7347926`, last accessed on 2023-09-24.

[JCQ] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics. version 8.0.0. 2023. URL: `https://github.com/ultralytics/ultralytics`, last accessed on 2023-09-24.

[JKSC20] Junho Jo, Hyung Il Koo, Jae Woong Soh, and Nam Ik Cho. Handwritten Text Segmentation via End-to-End Learning of Convolutional Neural Networks. *Multimedia Tools and Applications*, 79(43-44):32137–32150, August 2020.

[KDJ16] Praveen Krishnan, Kartik Dutta, and CV Jawahar. Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 289–294. IEEE, 2016.

[KFDS13] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting. In *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 560–564, Los Alamitos, CA, USA, aug 2013. IEEE Computer Society.

[KGS99] Gyeonghwan Kim, Venu Govindaraju, and Sargur N Srihari. An architecture for handwritten text recognition systems. *International Journal on Document Analysis and Recognition*, 2(1):37–44, 1999.

[KRW$^+$20] Lei Kang, Pau Riba, Yaxing Wang, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images. In *Computer Vision – ECCV 2020*, pages 273–289, Cham, 2020. Springer International Publishing.

[LB05] Marcus Liwicki and Horst Bunke. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 956–961 Vol. 2, 2005.

[LMB$^+$14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer International Publishing, 2014.

[MB02] Urs-Viktor Marti and Horst Bunke. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[MMB01] Urs-Viktor Marti, R. Messerli, and Horst Bunke. Writer identification using text line based features. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 101–105, 2001.

[Ots79] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[PPD$^+$21] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(3), 2021.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[RF18] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767, 2018.

[SLG78] Johansson Stig, Geoffrey N Leech, and Helen Goodluck. Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computers. Department of English, University of Oslo. 1978.

[SLY22] Qi Shen, Fangjun Luan, and Shuai Yuan. Multi-scale residual based siamese neural network for writer-independent online signature verification. *Applied Intelligence*, 52(12):14571–14589, March 2022.

[SRTV16] Joan Andreu Sanchez, Veronica Romero, Alejandro H Toselli, and Enrique Vidal. ICFHR2016 Competition on Handwritten Text Recognition on the READ Dataset. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 630–635. IEEE, 2016.

[TL19] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

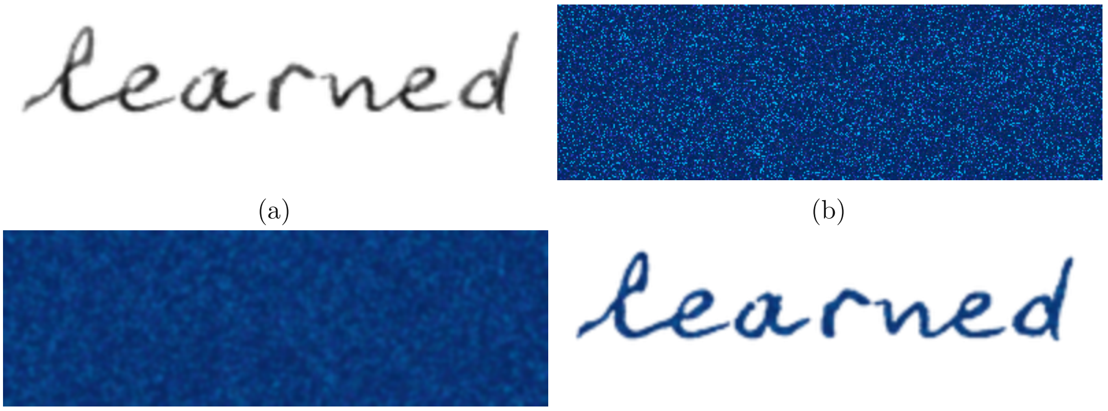
[Tol20] John Ronald Reuel Tolkien. *The fellowship of the ring*. The Lord of the Rings. HarperCollins, London, England, 2020.

[ZPA$^+$12] Konstantinos Zagoris, Ioannis Pratikakis, Apostolos Antonacopoulos, Basilis Gatos, and Nikos Papamarkos. Handwritten and Machine Printed Text Separation in Document Images Using the Bag of Visual Words Paradigm. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 103–108. IEEE, 2012.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

Appendix A: Stroke Color Augmentation

Stroke color augmentation is done by overlaying the HWT image with transparent foreground (i.e., the text) and white background over a randomly colored image. The latter is done by first creating an image with the same dimensions as the input data, but with three channels. The pixels within this image are randomly set with values of a particular color – e.g,. the first channel with the corresponding red value, the second with the blue value, and the third with the green value. A Gaussian filter with a kernel of 9x9 pixels is applied to this image, and the HWT image with transparent foreground is finally overlayed. Figure 1 visualizes this method. The random colors used to fill the image must all be from the same color shade. For example, only shades of blue are used to mimic a blue pen, only shades of red for a red pen, and so on.



(a)

(b)

(c)

(d)

Figure 1: An example for stroke color augmentation targeting a blue pen. (a) shows the input image whose stroke should be colored, the image is generated using [DMP+20]. (b) is an image with equal size as the input image which has been randomly filled with different shades of blue. (c) shows the randomly colored image after a Gauss filter with a 9x9 kernel has been applied. The final result is obtained by overlaying the input image having a transparent foreground over the filtered random image, as shown in (d).

Appendix B: Compute Maximum Rotation of Handwritten Text

Suppose two rectangles, where one rectangle has, at most, the dimensions (width and height) of the other; hence it is possible to place the smaller rectangle within the second one. The task is to find the maximum number of degrees the inner (i.e., the smaller) rectangle can be rotated around its center so that it is still enclosed by the outer rectangle, as shown in Figure 2. Denoting w_o, h_o, w_i, and h_i the width and height of the outer and inner rectangles, respectively, and r as half the diagonal of the inner rectangle, then the formula for computing the maximum degrees the inner rectangle can be rotated at without crossing the outer rectangle, α_{max}, is shown in Equation 1.

$$\begin{aligned}\alpha_{max} &= \alpha_{tot} - \alpha_{start} \\ &= \arcsin\left(\frac{\frac{h_o}{2}}{r}\right) - \arcsin\left(\frac{\frac{h_i}{2}}{r}\right) \\ &= \arcsin\left(\frac{\frac{h_o}{2}}{\sqrt{\left(\frac{w_o}{2}\right)^2 + \left(\frac{h_o}{2}\right)^2}}\right) - \arcsin\left(\frac{\frac{h_i}{2}}{\sqrt{\left(\frac{w_i}{2}\right)^2 + \left(\frac{h_i}{2}\right)^2}}\right) \\ &= \arcsin\left(\frac{h_o}{\sqrt{w_o^2 + h_o^2}}\right) - \arcsin\left(\frac{h_i}{\sqrt{w_i^2 + h_i^2}}\right)\end{aligned} \tag{1}$$

Notable is that Equation 1 is only valid if $h_o - h_i \leq w_o - w_i$ holds. In other words, the equation holds if the vertical margin between the inner and the outer rectangle is smaller or equal to the horizontal margin between the two rectangles. If the horizontal margin becomes smaller than the vertical margin, then h_o and h_i in the numerators of Equation 1 must be replaced with w_o and w_i, respectively – one can think of this as rotating the rectangles by 90° counterclockwise, then the heights of the rectangles become their widths.

For Equation 1, it is assumed that the inner rectangle is centered within the outer rectangle. Considering Figure 2, if the inner rectangle would be moved upwards, then the available space for rotation would be reduced; the same applies if the rectangle is moved

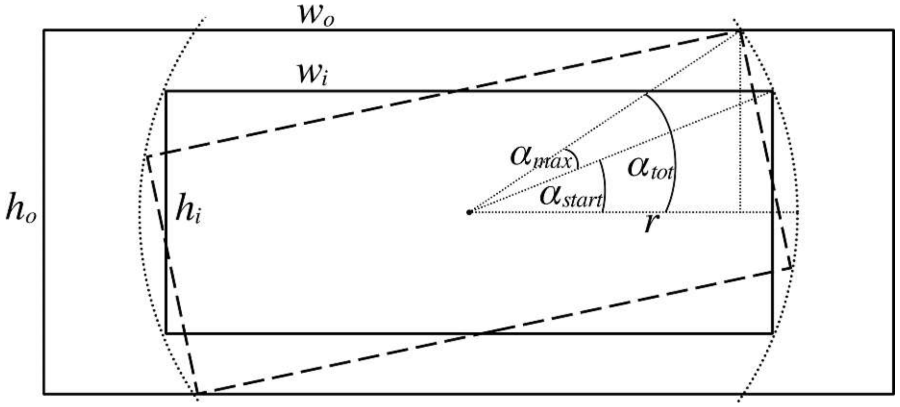


Figure 2: The width and height of the outer rectangle are w_o and h_o, respectively. The width and height of the inner rectangle are w_i and h_i, respectively r equals half the diagonal of the smaller rectangle. α_{start} is the angle at the beginning. α_{tot} is the maximum possible angle where, after rotation of the inner rectangle, it is still enclosed by the outer rectangle. α_{max} is the maximum possible angle the inner rectangle can be rotated by without crossing the borders of the outer rectangle. The dashed line represents the inner rectangle after being rotated by the maximum allowed angle; the dotted lines are helper lines for visualization.

downwards. Moving the inner rectangle to the left or the right would again decrease the available space, assuming the horizontal margin between the inner and outer rectangle is smaller than the vertical margin. Hence, centering the inner rectangle within the outer one maximizes the space for rotating the inner rectangle without crossing the borders of the outer one, which is why this assumption is made for Equation 1.

Appendix C: Finding Bounding Boxes for Words

The labels for HTD data sets must contain the positions of the HWT within the image. Computing those positions is straightforward for paragraphs, as it's known where they are placed within the image. Since the locations of lines within a paragraph are known, their absolute location within the image is also easily derived. However, the HWT synthesis model [DMP$^+$20] does not output any information about the position of single words within an HWT line. Hence, they have to be computed to generate labels on word-level.

This is done by identifying the $N-1$ largest whitespaces within an HWT line image, where N is the number of words within that image. Whitespaces are identified by computing the pixel-sums for each column of the binarized image (the background must be represented as 0, the HWT stroke as 1), all consecutive columns whose sum is 0 are considered as whitespace. This allows to identify the horizontal position of the words. A similar approach is applied to identify the vertical position. Figure 3 visualizes this method.

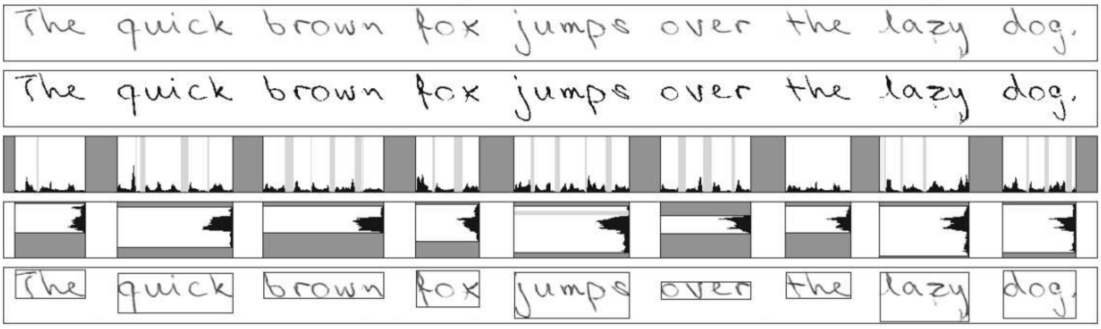


Figure 3: The first line shows the text “The quick brown fox jumps over the lazy dog”, which has nine words, on a synthetic HWT image generated using the model described in [DMP$^+$20]. The second line shows the binarized version using Otsu’s method [Ots79]. The gray areas in the third row mark those columns of the second image that do not contain HWT pixels. The outer two dark gray areas are the first and the last section without any HWT pixels, hence can be used to derive the beginning and end of the first and last word in the image, respectively. The remaining eight dark gray areas are the $N - 1 = 9 - 1 = 8$ widest consecutive sequences without any HWT pixels, hence can be used to derive the remaining start- and end-coordinates of the words. The black lines in the third row show the HWT pixel intensities for each column. The fourth line is similar to the third one, except only areas containing words are considered. I.e., the row segments are summed up, the gray areas indicate sequences without HWT pixels (only the lower and upper ones are of interest, filled with dark gray), and the black lines again show the HWT pixel intensities. The fifth line finally shows the derived bounding boxes.

Appendix D: Training Parameters for Object Detection Models

For YOLOv5 models, the default hyper parameters for low-augmentation training are used[1]:

- Optimizer: Stochastic Gradient Descent (SDG)
- Initial learning rate: 0.01
- Final OneCycleLR learning rate (lr0 * lrf): 0.01
- SGD momentum: 0.937
- Optimizer weight decay: 0.0005
- Warmup epochs: 3.0
- Warmup initial momentum: 0.8
- Warmup initial bias lr: 0.1
- Box loss gain: 0.05
- Cls loss gain: 0.5
- Cls BCELoss positive weight: 1.0
- Obj loss gain (scale with pixels): 1.0
- Obj BCELoss positive weight: 1.0
- IoU training threshold: 0.20
- Anchor-multiple threshold: 4.0

[1] The descriptions are taken from `https://github.com/ultralytics/yolov5/blob/master/data/hyps/hyp.scratch-low.yaml`, last accessed on 2023-09-12.

- Anchors per output layer: 3.0
- Focal loss gamma: 0.0
- Image HSV-Hue augmentation (fraction): 0.015
- Image HSV-Saturation augmentation (fraction): 0.7
- Image HSV-Value augmentation (fraction): 0.4
- Image rotation (+/- deg): 0.0
- Image translation (+/- fraction): 0.1
- Image scale (+/- gain): 0.5
- Image shear (+/- deg): 0.0
- Image perspective (+/- fraction), range 0-0.001: 0.0
- Image flip up-down (probability): 0.0
- Image flip left-right (probability): 0.5
- Image mosaic (probability): 1.0
- Image mixup (probability): 0.0

For YOLOv8 models, following hyper parameters are used[2]

- Optimizer: Stochastic Gradient Descent (SDG)
- Initial learning rate: 0.01
- Final learning rate: 0.01
- SGD momentum: 0.937
- Optimizer weight decay: 0.0005
- Warmup epochs: 3.0
- Warmup initial momentum: 0.8
- Warmup initial bias lr: 0.1
- Box loss gain: 7.5

[2] The descriptions are taken from `https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/default.yaml`, last accessed on 2023-09-12.

- Cls loss gain (scale with pixels): 0.5
- Dfl loss gain: 1.5
- Keypoint obj loss gain: 1.0
- Label smoothing (fraction): 0.0
- Image HSV-Hue augmentation (fraction): 0.015
- Image HSV-Saturation augmentation (fraction): 0.7
- Image HSV-Value augmentation (fraction): 0.4
- Image rotation (+/- deg): 0.0
- Image translation (+/- fraction): 0.1
- Image scale (+/- gain): 0.5
- Image shear (+/- deg): 0.0
- Image perspective (+/- fraction): 0.0
- Image flip up-down (probability): 0.0
- Image flip left-right (probability): 0.5
- Image mosaic (probability): 1.0
- Image mixup (probability): 0.0